DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Computations and Algorithms in Physical and Biological Problems

Citation
Qin, Yu. 2014. Computations and Algorithms in Physical and Biological Problems. Doctoral dissertation, Harvard University.

Permanent link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:12274619

Terms of Use
This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Computations and Algorithms in Physical and Biological Problems

A DISSERTATION PRESENTED
BY
YU QIN
TO
THE SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
APPLIED MATHEMATICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
APRIL 2014

©2014 Yu Qin
ALL RIGHTS RESERVED.

Dissertation Adviser: Professor Michael P. Brenner Yu Qin

Computations and Algorithms in Physical and Biological Problems

Abstract

This dissertation presents the applications of state-of-the-art computation techniques and data analysis algorithms in three physical and biological problems: assembling DNA pieces, optimizing self-assembly yield, and identifying correlations from large multivariate datasets. In the first topic, in-depth analysis of using Sequencing by Hybridization (SBH) to reconstruct target DNA sequences shows that a modified reconstruction algorithm can overcome the theoretical boundary without the need for different types of biochemical assays and is robust to error. In the second topic, consistent with theoretical predictions, simulations using Graphics Processing Unit (GPU) demonstrate how controlling the short-ranged interactions between particles and controlling the concentrations optimize the self-assembly yield of a desired structure, and nonequilibrium behavior when optimizing concentrations is also unveiled by leveraging the computation capacity of GPUs. In the last topic, a methodology to incorporate existing categorization information into the search process to efficiently reconstruct the optimal true correlation matrix for multivariate datasets is introduced. Simulations on both synthetic and real financial datasets show that the algorithm is able to detect signals below the Random Matrix Theory (RMT) threshold. These three problems are representatives of using massive computation techniques and data analysis algorithms to tackle optimization problems, and outperform theoretical boundary when incorporating prior information into the computation.

Contents

I dedicate this dissertation to my fiancee Yang Cao, who always selflessly supports me and brings me sunshines and smiles everyday.

Acknowledgments

First and foremost, it is with immense gratitude that I acknowledge the support and help of my thesis adviser, Professor Michael Brenner, who has the attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this dissertation would not have been possible.

I share the credit of my work with many colleagues, particularly, Dr. Tobias Schneider who jointly contributes to the sequencing project discussed in Chapter 1, Dr. Zorana Zeravcic and Dr. Arvind Murugan who work closely with me on the self-assembly project elaborated in Chapter 2, and Dr. Lucy Colwell who provides insightful ideas on Random Matrix Theory presented in Chapter 3. It is your inspiring and cheerful conversations that guide me step-by-step towards the completion of this dissertation.

I am indebted to my beloved family and friends. In particular, I owe my deepest gratitude to my parents, who constantly support me through my graduate school, and my fiancee Yang Cao, who has been bringing me countless happiness and sunshine. This dissertation would have remained a dream had it not been for all your loves.

I would also like to thank two Harvard alumni, Dr. Cheng Chi Kao and Dr. Herbert S. Winokur, who kindly support my graduate study through Kao Fellowship and Herbert S. Winokur SEAS Fellowship. Last but not least, I wish to thank Darryl Zeigler at the Harvard International Office, who truly understands the hardness of foreign students from different cultures seeking educations in the US, and sincerely supports them as much as he can.

Things don't have to change the world to be important.

Steve Jobs

0 Introduction

Scientific computing[60,46] is ubiquitous in physical[36,17], biological[48,63] and engineering sciences[58,78]. Recent advances in High Performance Computing (HPC)[42,99] open new opportunities to tackle analytically complicated scientific problems, such as those in astrophysics and turbulence[65,44], with modern computation techniques. Due to the proliferation of data collection in almost every area of science, the enormous datasets now routinely encountered in sciences, or the so-called *Big Data*[66], provide another

incentive to rely on computations, as well as its marriage with data analysis algorithms, to synthesize, interpret and give meaning to the data in the context of its scientific setting. In this dissertation, I present the applications of state-of-the-art computation techniques and data analysis algorithms in three representative physical and biological problems: assembling DNA pieces, optimizing self-assembly yield, and identifying correlations from multivariate datasets.

Chapter 1 is about the reconstruction algorithm in Sequencing by Hybridization (SBH)[13,30,29,57]. SBH uses the binding characteristics of a library of short DNA probes, or *oligonucleotides*, to reconstruct a target DNA sequence. Traditionally, SBH has been carried out using microarrays[27,41], but recent advances in microfluidics have created the possibility of carrying out the hybridization reactions inside small droplets in high throughput[6,5,gnu]. This creates new opportunities for using SBH creatively for sequencing. The reconstruction algorithm after collecting data from hybridization experiment is a critical part of SBH. Existing Algorithms[33,18,22] have been focusing on accommodating biochemical errors. However, in these approaches, the length n of a uniformly random sequence that can be unambiguously reconstructed with probes of length l is limited to $n = \mathcal{O}(2^l)$ due to repetitive subsequences causing reconstruction degeneracies[31,11], while increasing the probe length exponentially increases the probe library size, and so forth the sequencing cost. This degeneracy essentially dims SBH from the fast growing sequencing industry. In our research, we however discover that although ambiguity emerges when sequencing long targets, there are finite number of possible reconstructed sequences which can be exhaustively enumerated, and incorporating some existing information later can identify the unique solution from a candidate solution set. Taking advantage of this discovery, this chapter presents a *swarm intelligence*[23] based SBH reconstruction algorithm[87] that overcomes the theoretical

boundary without increasing the probe length or the need for different types of biochemical assays, and is robust to error. It opens the potential to sequence long DNAs with SBH.

Chapter 2 discusses optimizations in self-assembly[100,101,102], a process where simple building blocks spontaneously assemble into structures of higher complexity. Recent experimental advances have opened up the possibility of equilibrium self-assembly of functionalized nanoblocks with a high degree of controllable specific interactions[88,39,81,96]. In recent years, researchers have been actively searching for design methods to optimize the assembly yield of specific structures[50,43,68], with the hope that one day we can manufacture large structures in batch with self-assembly. In this chapter, in light of examples from colloidal engineering where particles interact in short range[69], I explore two self-assembly optimization methods using massive Dissipative Particle Dynamics (DPD)[49,40] simulation: controlling the short-ranged interactions between particles[50], or *coloring*, and controlling the concentrations of different types of particles in the system. In particular, a state-of-the-art parallel computing technique, the General-Purpose Graphics Processing Unit (GPU)[75,79,80,24], is applied to achieve efficient computation when simulating particle systems with numerous particles. As a result, a large particle system reaches equilibrium in significantly shorter computation time than its serial counterpart. Results show that the yield optimization are consistent with theoretical predictions[50,70]. Leveraging the enormous computation capacity of GPUs, we are also able to uncover the nonequilibrium, or finite time behavior of a particle system when optimizing concentrations. This is achieved by simulating a large number of independent particle systems in parallel and aggregating data from all systems.

Chapter 3 presents an effort of identifying correlations from large multivariate

datasets. Technological advances enable the measurement of an ever increasing number of variables during an experiment. In biological settings this occurs, for example, when measuring the expression levels of all genes in a cell at different time points or when comparing gene expression levels between individuals[85,7,47]. Examples from other fields include the time series of stock prices[86,61], weather patterns[90], and the study of networks[71,72]. In these settings, the number of variables p is comparable to, or even larger than the number of samples n. A critical question when analyzing such multivariate datasets is to understand the correlations between variables, and their clustering so forth. A common technique for identifying true correlations from sample correlation matrix is the Principal Component Analysis (PCA)[54,53], where the largest eigenvalues of the sample correlation matrix are assumed to correspond to true correlations between sets of variables rather than spurious correlations caused by sampling noise. The Random Matrix Theory (RMT)[12,32,52], on the other hand, provides a set of methodologies to facilitate interpretations of these eigenvalues, including determining a threshold to seperate informative eigenvalues and noise. This chapter presents an algorithm based on *simulated annealing*[59,97] to efficiently traverse the *likelihood* landscape and search for the optimal true correlation matrix of a particular pattern. Further comparison shows that its signal detection power is equivalent to RMT's; however, when some existing information is incorporated into the algorithm, it is capable to identify signals below the RMT threshold. The efficacy of the algorithm is tested on both synthetic and real financial datasets.

The above mentioned three topics are superficially independent, yet intrinsically they share the same spirit of leveraging computation and data analysis techniques to tackle optimization problems, and outperform theoretical boundary when incorporating prior knowledge into the computation. The next three chapters elaborate these

three problems one by one. Chapter 4 Concludes the dissertation.

Learn from yesterday, live for today, hope for tomorrow.

The important thing is not to stop questioning.

Albert Einstein

1

Breaking the Boundary of Sequencing by Hybridization

Sequencing by Hybridization (SBH)[13,30,29,57] reconstructs an n-long target DNA sequence from its biochemically determined l-long subsequences. In the standard approach[84], the length of a uniformly random sequence that can be unambiguously reconstructed is limited to $n = \mathcal{O}(2^l)$ due to repetitive subsequences causing recon-

struction degeneracies[31,11]. In this chapter, I present a modified sequencing method[87] that overcomes this limitation without the need for different types of biochemical assays and is robust to error.

The organization of the chapter is as follows. Section 1.1 gives an overview of SBH and its experimental methods; Section 1.2 introduces the reconstruction degeneracy of SBH; Section 1.3 presents the graph based reconstruction model which can accommodate moderate error rates; Section 1.4 thoroughly discusses the cause of the reconstruction degeneracy; Section 1.5 presents a modified reconstruction algorithm that overcomes the degeneracy; Section 1.6 shows simulation results on both random and natural DNA sequences, and Section 1.7 summarizes the chapter.

1.1 SBH Introduction

Figure 1.1 shows a schema of the SBH procedure. A single-strand target sequence is tested with each probe in a probe library, all 3-mers as an example, for hybridization possibilities. When a probe binds to the target, it means that its contrary exists as a subsequence of the target. Then we collect all probes that bind to the target, denoted as the *spectrum*, and use their overlap characteristics to reconstruct the target. Figure 1.2 shows two methods to carry out the SBH. Classically, SBH is carried out on microarray [27], where hybridization reactions between the target and different probes are performed at many microscopic DNA spots. Recent works in microfluidics, however, open the possibilities to wrap the target and probes inside microdroplets, and the hybridization reactions are triggered when two droplets coalescence [6,5,gnu]. This technology could revolutionize SBH, and even the DNA sequencing industry, as the its cost to perform sequencing is dramatically lower than classical methods. Therefore, it is valuable to revisit the reconstruction algorithm of SBH.

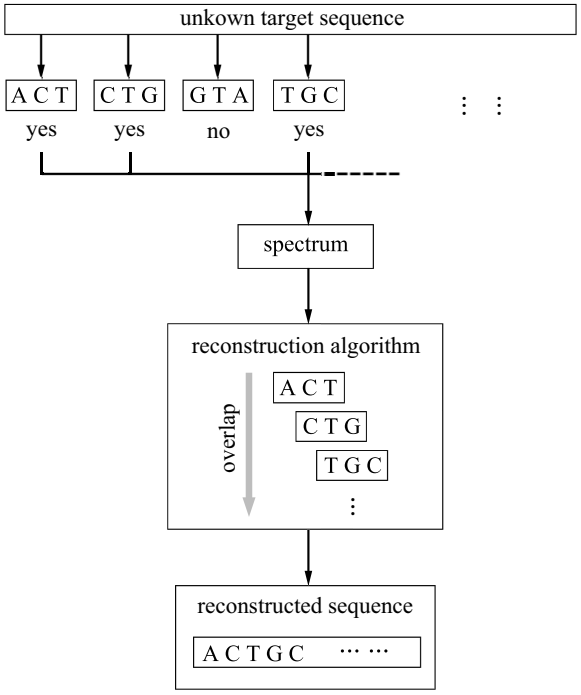


Figure 1.1: A schema of the SBH procedure. A single-strand target sequence is tested with each probe in a probe library, all 3-mers as an example, for hybridization possibilities. When a probe binds to the target, it means that its contrary exists as a subsequence of the target. Then we collect all probes that bind to the target, denoted as the *spectrum*, and use their overlap characteristics to reconstruct the target.

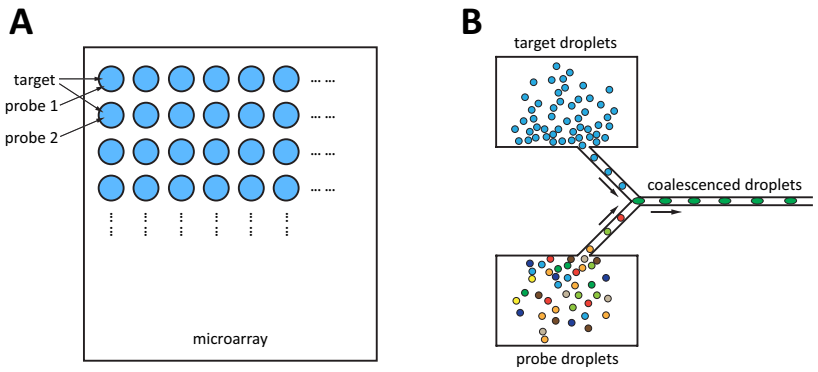


Figure 1.2: Two methods to carried out SBH: the microarray method (**A**) and the microfluidic method (**B**). In the microarray method, hybridization reactions between the target and different probes are performed at many microscopic DNA spots. In the microfluidic method, target and probes are wrapped inside microdroplets, and the hybridization reactions are triggered when two droplets coalescence. The microfluidic method is significantly more efficient and economic than the microarray method. It opens the opportunities to revolutionize the DNA sequencing industry.

1.2 SBH Limitations

A fundamental limit to the length n of the target sequence that can be sequenced by probes of length l follows from an information theoretic bound: since there are 4^l probes in a standard probe library, each of which may or may not bind to a subsequence of the target, the probe library can give 2^{4^l} possible measurements; comparing this with the 4^n possible target sequences implies that for a unique measurement to be associated with every possible target sequence, we need $2^{4^l} \geq 4^n$, or $n \leq 4^l/2$. Hence, for probes of length $l = 7$, the maximum target sequence length is $n = 4^7/2 = 8192$.

In fact, the maximum length of the target that can be sequenced is much below the above threshold due to repetitive subsequences. If, for example, two oligonucleotides in the spectrum, b and c, have their $(l-1)$-mer prefixes both identical to the $(l-1)$-mer suffix of a third oligonucleotide a, then there are *two* possible target sequences, with either b or c as the successor of a, that are consistent with the probe binding character-

istics. Analysis of uniformly random target sequences shows that this results in an SBH reconstruction boundary $n \leq 2^{l}$ [31,11], reducing the maximum length of a target that can be sequenced with probes of length 7 to 128. The dramatic decrease in the length of sequenceable targets has severely limited the efficacy of SBH [84]. To increase the length of sequenceable targets while fixing the size of the probe library, theoretical concepts have previously revolved around using optimized probe patterns involving non-specifically binding universal bases instead of standard oligonucleotide probes [37], which have been proven hard to implement.

1.3 Biochemical Errors and Graph Based Reconstruction

An additional challenge is biochemical errors in measuring the probe spectrum, including *positive errors* and *negative errors*. A common approach to accommodate moderate error rates is to model the SBH reconstruction as an optimization problem in a completely connected directed graph[19]. As a simple example, suppose we sequence a target sequence

$$t = ACTGACTC \tag{1.1}$$

with probes of length $l = 3$. The ideal spectrum is

$$S_i = \{ACT, CTG, TGA, GAC, ACT, CTC\}. \tag{1.2}$$

Since we can only read out a set of yes-or-no answers in the experiment, we will miss the duplicated ACT. Besides, suppose there is a positive error TAA and a negative error CTG, then the actual spectrum from the experiment is

$$S_a = \{ACT, TGA, GAC, CTC, TAA\}. \tag{1.3}$$

We can build a completely connected directed graph based on the actual spectrum, where vertices represent probes in S_a, and weights of the directed edges are the number of overlapping bases between the suffix of the starting probe and the prefix of the ending probe, as Figure 1.3A shows. The reconstruction of the target is then equivalent to seeking an optimal path p^* in the graph:

$$p^* = \text{argmax}_p[\,\text{len}(p)\,], \quad \text{s. t. } \text{len}[\,r(p)\,] = l \times \text{len}(p) - \sum_{i=1}^{\text{len}(p)-1} o_{p_i p_{i+1}} \leq n, \tag{1.4}$$

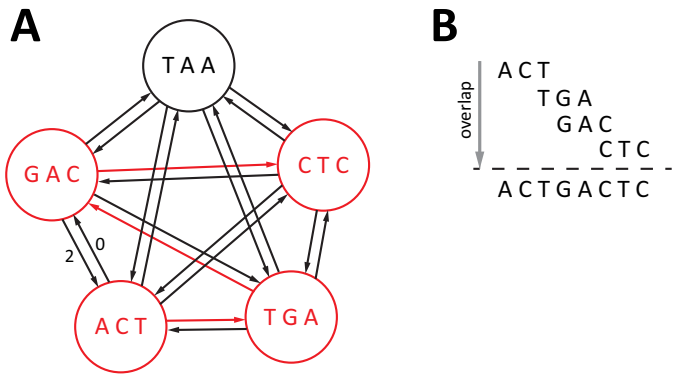


Figure 1.3: Graph based reconstruction. SBH reconstruction is modeled as an optimization problem in a completely connected directed graph, where vertices represent probes in the actual spectrum, and weights of the directed edges are the number of overlapping bases between the suffix of the starting probe and the prefix of the ending probe. **A** shows an example graph when reconstructing a target ACTGACTC with 3-mers, and there is one positive error TAA and one negative error CTG in the actual spectrum. Color red highlights the optimal path of the graph obtained by the optimization (1.4). **B** shows the reconstruction using the overlap characteristics of probes in the optimal path.

where p is a path in the graph; $\text{len}(p)$ is the number of vertices in the path; $\text{len}[\, r(p)\,]$ is the length of the reconstructed sequence; p_i is the ith vertex of p; n is the target sequence length; l is the probe length, and o_{ij} is the number of overlapping bases from vertex i to j. We can algorithmically determine the starting vertex of the path following a two-step procedure: first, find a set of vertices $S_{bs} \subseteq S_a$, such that the highest-weighted outgoing edge of each vertex in S_{bs} is greater than or equal to the highest-weighted outgoing edges of all other vertices in S_a; second, find a set $S_{wp} \subseteq S_{bs}$, such that the highest-weighted incoming edge of each vertex in S_{wp} is less than or equal to the highest-weighted incoming edges of all other vertices in S_{bs}. The vertex in S_{wp} is used as the starting point of the path. If there are more than one vertex in S_{wp}, one of them is chosen randomly. In Figure 1.3A, color red highlights the optimal path of the graph obtained by the above optimization (1.4). Figure 1.3B shows the reconstruction using the overlap characteristics of probes in the optimal path.

It has been proved that SBH with errors is NP-hard [21]. Therefore, when the number of vertices becomes large, it is unpractical to solve the above optimization problem with exhaustive search. A number of heuristic algorithms has been developed [33,18,22]. The simplest heuristics is the greedy algorithm, which always follows the outgoing highest-weighted edge in each searching step after the first vertex is determined. However, this method is frail to biochemical error, and moreover, it will certainly encounter ambiguity at the search step where the number of outgoing highest-weighted edges is more than one. In later simulations, we use a more intelligent heuristics, the Ant Colony Optimization (ACO) algorithm [22]. It can overcome moderate positive and negative errors and perform close to the intrinsic reconstruction boundary, or $n = 2^l$.

The ACO imitates the mechanism of a swarm of ants searching for food. Ants deposit *pheromone* on the path they pass to guide successive fellow ants. Eventually, the swarm of ants will converge to the shortest path between the nest and the food source, as the amount of pheromone on this path grows to become the largest. In ACO, the algorithm searches path in the graph for many iterations, indexed by x. The edge from vertex i to j is not only weighted by the directed overlap o_{ij}, but also by a pheromone value $\tau_{ij}(x)$. The pheromone values of all edges are initialized to the same value $\tau_{ij}(0)$. The algorithm then keep updating τ_{ij} in each iteration according to a specified rule. The weighting function μ_{ij} on edge from i to j is given by

$$\mu_{ij} = (\frac{o_{ij}}{l-1})^5 \tau_{ij}(x). \tag{1.5}$$

The algorithm uses an nondeterministic strategy in each search step: for a user-specified probability $q \in [0, 1)$, the next edge is chosen to be the highest-weighted outgoing edge; otherwise the next edge is randomly chosen from a candidate list of a user-

specified number of top-weighted outgoing edges. This strategy gives the search opportunities to switch from the highest-weighted but incorrect edge to less-weighted but correct edge. Therefore, the ACO is relatively robust to error and ambiguity.

In this chapter, we present a methodology that overcomes the problem of degeneracy due to multiple repeats, is robust to errors and only requires the use of standard oligonucleotide probes. The idea is to use the fact that although repeats cause the probe binding characteristics to correspond to multiple possible target sequences, the set of possibilities for a target sequence can be completely enumerated. By randomly fragmenting multiple copies of the target sequence the resulting probe binding data of the fragments can be combined to uniquely identify the target sequence. Numerical simulations demonstrate that this sequencing method outperforms the classical $n = 2^l$ boundary for random sequences, has excellent performance on natural sequences, and is robust to error.

1.4 A Closer Look at Failures Due to Subsequence Repeats

Reconstructing a target sequence from its subsequences in the presence of errors is a computationally complex problem [21], requiring the use of heuristic search algorithms which randomly find one of the possibly multiple solutions. We implement a reconstruction algorithm based on the ant colony optimization (ACO) [22], and demonstrate that the failure of the classical SBH method is indeed correlated with the structure of subsequence repeats.

Using the $l = 10$-mer spectrum, we simulate the reconstruction of natural DNA sequences of length $n = 509$, taken from a benchmark library [20*]. Figure 1.4 shows 10 independent simulated reconstruction attempts of three representative instances of the benchmark with random 1% positive and 1% negative errors added to each spectrum. Visualizing the repeat structure of the target DNA demonstrates that the existence of multiple solutions with degenerate spectra requires at least two pairs of 9-mer repeats, i.e. repeats one base shorter than the probes, arranged in an appropriate configuration. As a result, the single pair of 9-mer repeats in Figure 1.4A does not lead to any reconstruction errors, nor does the double pair of 9-mer repeats in Figure 1.4B, where the second pair of repeats directly follows the first. In contrast, Figure 1.4C shows that if the 9-mer repeats interlace each other, half of the reconstructions have significant reconstruction error of the same pattern right after a 9-mer repeat, but switches back to the correct solution exactly at the beginning of a second 9-mer repeat. This 'wrong' reconstruction is actually a second solution compatible with the same spectrum. These rules persist in our reconstructions of the entire benchmark library (see the online applet [vis] and instructions in the supporting information).

*The library that contains 10-mer repeats.

A symbolic representation further demonstrates that both the number of $(l-1)$-mer repeats and their aligning pattern are part and parcel for the non-uniqueness of solutions. Suppose we have a sequence with two pairs of $(l-1)$-mer repeats, with subsequences denoted by A_1 and A_2, respectively. If the target sequence is $BA_1CA_2DA_1EA_2F$, with B, C, D, E and F arbitrary (nonrepeating) subsequences, then another possible reconstruction of this sequence is $BA_1EA_2DA_1CA_2F$. Note that this second reconstruction, which switches the subsequence A_1CA_2 with A_1EA_2, has exactly the same spectrum as the target sequence. In contrast, if two pairs of repeats are arranged in the target sequence as $BA_1CA_1DA_2EA_2F$, then there is no rearrangement of the subsequences that leads to a reconstructed sequence with the same spectrum as the target. In general, given repeats A_i and A_j, if there exist two subsequences of the target sequence of the form A_iBA_j and A_iCA_j, then interchanging these subsequences leads to a reconstruction with a consistent spectrum. In the case of a triple repeat, i.e. a target sequence of the form $BA_1EA_1DA_1C$, a second sequence which matches the spectrum is $BA_1DA_1EA_1C$.

1.5 Methods

Although a target sequence with a repeat structure that leads to multiple reconstructions cannot be uniquely identified with a single round of SBH, the set of target sequences compatible with the spectrum is finite. We now show that fragmenting multiple copies of a long target sequence and enumerating the complete sets of possible sequences for each fragment allows the unique identification of the target. Since all fragments have to be compatible with the same long target DNA, there is sufficient additional information to break the degeneracy, choose a specific fragment sequence and uniquely reconstruct a uniformly random target for $n > 2^l$. The proposed method is summarized in Figure 1.5.

1.5.1 Solution enumeration

To enumerate all possible fragment sequences consistent with a given probe spectrum, we first determine one solution using a standard reconstruction algorithm. Figure 1.6 shows how the enumeration is done for a sequence containing a triple $(l-1)$-mer repeat. After detecting the locations of the repeats, the algorithm starts from the beginning of the known solution and enumerates all possible extensions after the first repeat. Continuing along the sequence, the process of enumerating all possible extensions and permuting parts of the sequence accordingly is iterated. The search is terminated when it reaches the end of the known solution, and we discard any search that does not cover the full length. Since an exhaustive search over all possible permutations of the $(l-1)$-mer repeats is performed, the set of solutions is complete.

1.5.2 Unique reconstruction

Since all fragments stem from the *same* sequence, we can uniquely determine the target by choosing specific solutions for the individual fragments. In order to find a target sequence consistent with all the candidate sets we use a variant of ACO, which starts by randomly selecting the solution sets of some fragments. For each candidate in these sets, the algorithm iteratively determines its left or right successors by heuristically choosing a candidate from one of the remaining candidate sets. The optimum is the reconstruction of the target sequence.

1.6 Simulation Results

1.6.1 Testing the method on random sequences

To test the method, we carry out simulations on 20 randomly generated 5000-mers, and attempt to sequence them using probes of length 7. We replicate each 5000-mer 8 times, randomly separate each replica into fragments of length $180 \leq n \leq 220$, whose locations in the target are unknown, and include 5% positive and 5% negative errors into the spectrum of each fragment. Therefore, roughly $5000/200 \times 8 = 200$ fragments need to be sequenced for each 5000-mer. Note that by choosing an average fragment length of 200 we are well above the traditional boundary for SBH, since $2^7 = 128$. In the assembly, we begin with 8 randomly selected solution sets. Figure 1.7A shows the performance comparison between the proposed method (blue circles) with a control method (red squares). The proposed method has an average similarity score of 94.4% over the 20 trials, including 2 accidental drops which are presumably due to errors in the spectra, whereas the control method has an average similarity score of 33.7%. In contrast, Figure 1.7B shows that only about half (45.3% on average) of the fragments are correctly reconstructed. Figure 1.7C and Figure 1.7D illustrate the number of candidate solutions for fragments used in the assembly. The average number of solutions ranges from 2 to 10, whereas the maximum exceeds 300. Despite these degeneracies, the accuracy of the reconstruction of the entire 5000-mer is nearly perfect.

1.6.2 Testing the method on natural sequences

The repeat distribution of natural sequences is not random, and hence, when sequencing with probes of length l, the expected length of a fragment for non-unique solu-

tions will be different than 2^l. The present method will be useful when the fragment length is sufficiently long to have multiple interlacing repeats in the sequence, but not too long that prohibitively many replicas are needed to be analyzed to break the degeneracy. Unlike the case of random sequences, we do not know this length *a priori*, therefore we need to find it as part of the reconstruction procedure.

To verify this, we carry out simulations on the 5000-mer prefix of three natural sequences from human RNA and bacterial DNA respectively (GenBank accession numbers JA638618, AEQT01000438 and AFZZ01000001). In the absence of detailed information about the statistics of their repeats as a priori, we empirically change the fragment length n by drawing it randomly from intervals ranging from $[90, 110]$ to $[190, 210]$, i.e. from $[100 - 10, 100 + 10]$ to $[200 - 10, 200 + 10]$, with step 10[†]. We replicate each target 10 times, and sequence them with 7-mers. 5% positive and 5% negative errors are added to the spectrum of each fragment. Figure 1.8A, B and C respectively shows the average similarity score over ten independent reconstruction attempts, using both the proposed method (blue circles) and the control method (red squares), as a function of the fragment length for the three 5000-mers. Figure 1.8D shows the average number of fragments need to be sequenced for different fragment length. With error bars representing the standard deviation, we can infer that the optimal fragment length for the sequences in Figure 1.8A and B are roughly 150 and 170, respectively, where sequencing 350 and 310 fragments give the proposed method larger than 95% in similarity scores, which significantly outperforms the control method. Note that the optimal fragment length increases when sequencing with more replicas. For the sequence in Figure 1.8C, the optimal fragment length is unclear when sequencing with 10 replicas;

[†]From then on we will use the center of the interval as a representative when describing fragment length.

however, we can still see the significant performance gain for the proposed method. Since different natural sequences have different optimal fragment length when separating replicas of the target, the optimal length of a target must be determined adaptively[‡]. The computation time varies over sequences, since it depends on both the number and the aligning pattern of $(l-1)$-mer repeats in the target sequence at hand. For $n = 150$, the average MATLAB computation time for the three 5000-mers are 1459s, 690s and 736s respectively on a Mac Pro with two 2.26GHz Quad-Core Intel Xeon processors.

The visualization in Figure 1.9 further demonstrates the efficacy of our sequencing method. It illustrates one simulation attempt of the human RNA sequence (JA638618) when using the optimal fragment length ($n = 150$) to separate replicas. Each row represents one replica of the 5000-mer. Although only 40.5% fragments are correctly reconstructed before solution enumeration (green bars), fragments that are not correctly reconstructed but have the correct solution in their candidate solution sets (blue bars) help to bridge gaps in the final assembly step, leading to near-perfect reconstruction.

[‡]For example, we can sequentially separate one replica, analyze its repeat pattern, and adaptively determine a better fragment length to separate the next replica.

1.7 Summary

To summarize, this chapter has demonstrated how a variant of the classical SBH algorithm can significantly extend the length of target that can be sequenced with standard oligonucleotide probes. The simulations indicate that both random and natural sequences of length 5000 can be accurately sequenced with standard 7-mer probes, even in the presence of 5% positive and negative errors. This is significantly longer than the theoretical boundary of SBH. The proposed algorithm creates new opportunities to use SBH along with the state-of-the-art microfluidic technology in the next generation sequencing.

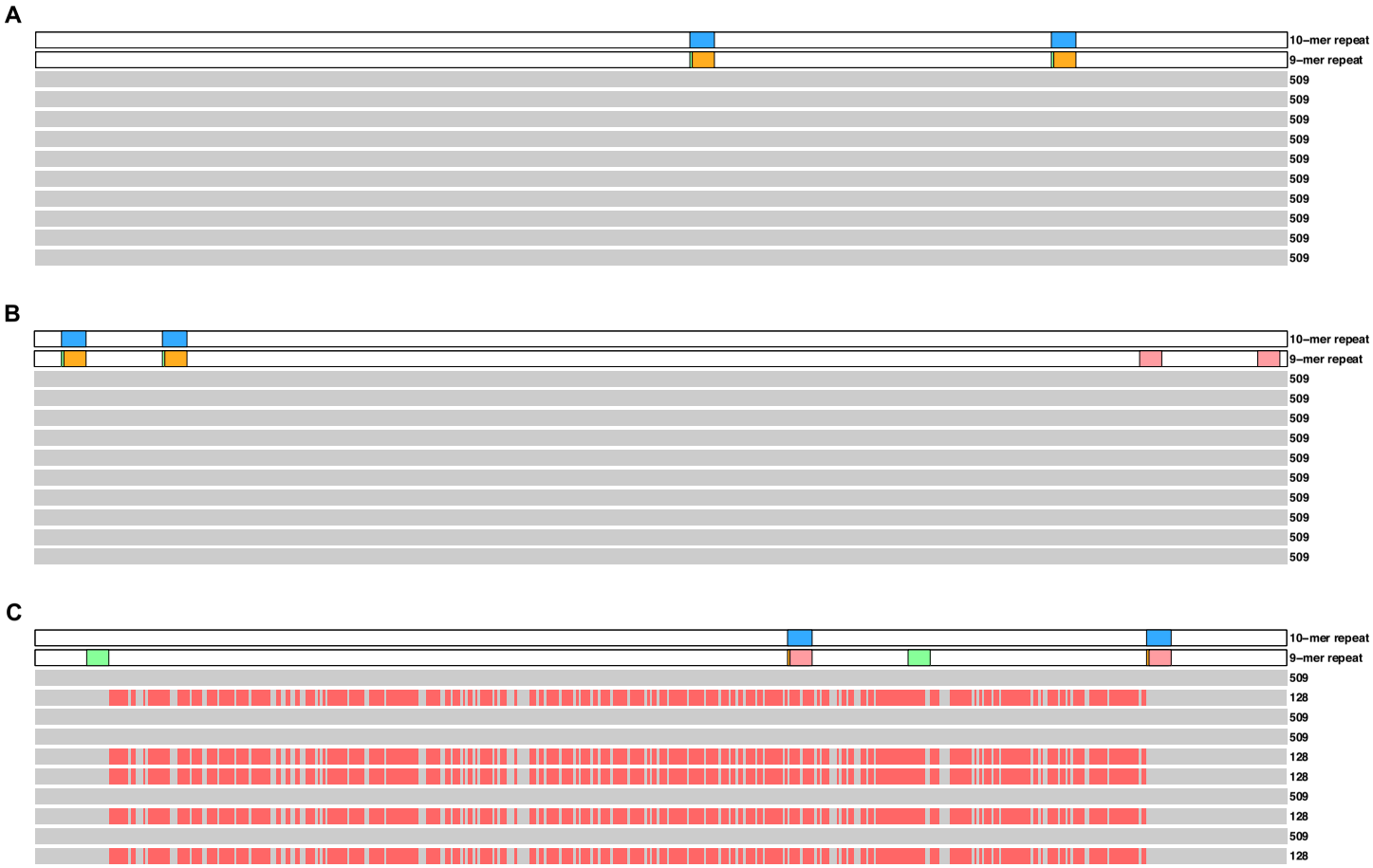


Figure 1.4: Sequencing failure due to degenerate solutions. Classical sequencing by hybridization of three representative (No. 21, 3 and 20) 509-mer target sequences taken from a benchmark library [20] using 10-mer probes. For each target sequence, 10 independent reconstruction attempts using a state-of-the-art heuristic algorithm (ACO [22]) are simulated and the results are visualized with grey (pink) color representing bases that are correctly (incorrectly) matched by the reconstruction. Reconstruction failures are correlated with the locations of 9-mer but not with 10-mer repeats within the target sequence, shown in the upper two rows of each figure. For two interlaced 9-mer repeats (C), two different solutions with Needleman-Wunsch similarity score [73] between the target and the reconstruction of 509 and 128 are found. Both sequences have the same 10-mer probe spectrum and can thus not be distinguished using classical SBH. For reconstruction attempts of other target sequences see the online applet [vis] and instructions in the supporting information.

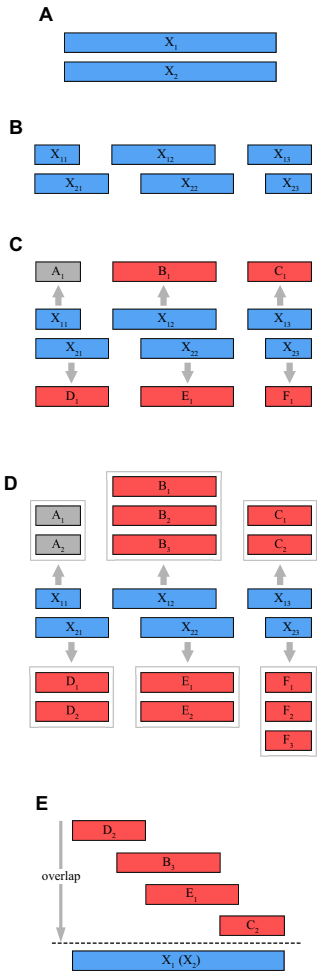


Figure 1.5: Schematic of the proposed sequencing method. The target sequence X is replicated multiple times (A) and the different replicas $X_i = X$ with $i = 1, 2$ are randomly separated into fragments X_{ij} (B). Sequencing each fragment using classical SBH yields one possible 'candidate' $A_1, B_1, \ldots$ for each fragment's sequence (C). Using the repeat structure of the candidate, the complete set of possible fragment sequences is constructed (D). Aligning the candidates allows to uniquely determine the target sequence and choose one element of each candidate set (E). The method is robust against erroneous candidate sets resulting from errors in the determined probe spectra of fragments, indicated as grey bars in (D).

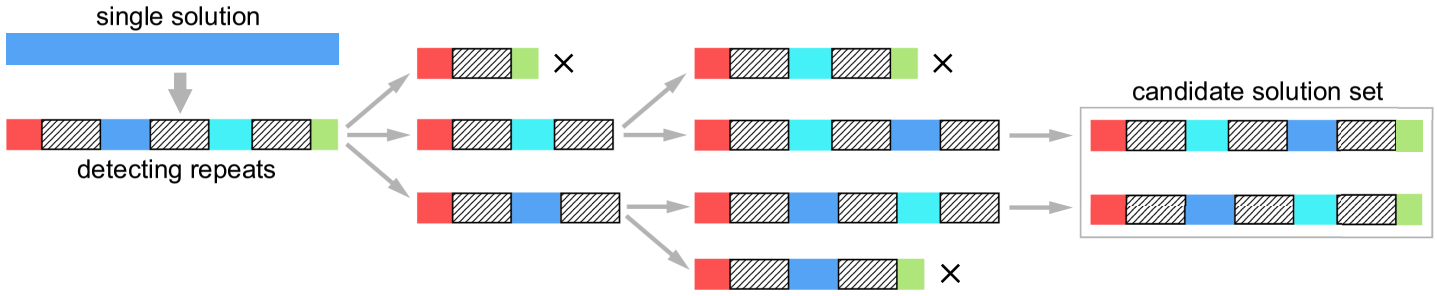


Figure 1.6: Constructing the complete candidate set. The algorithm first detects the locations of (l-1)-mer repeats in a single solution (leftmost column, repeats are shaded grey). Then we enumerate all possible extensions of the repeat (2nd column), continuing until we arrive at a set of candidate solutions that have the same length as the original sequence (3rd and 4th columns). Since all permutation of the subsequences between repeats are checked for consistency with the probe spectrum, all degenerate solutions are found.

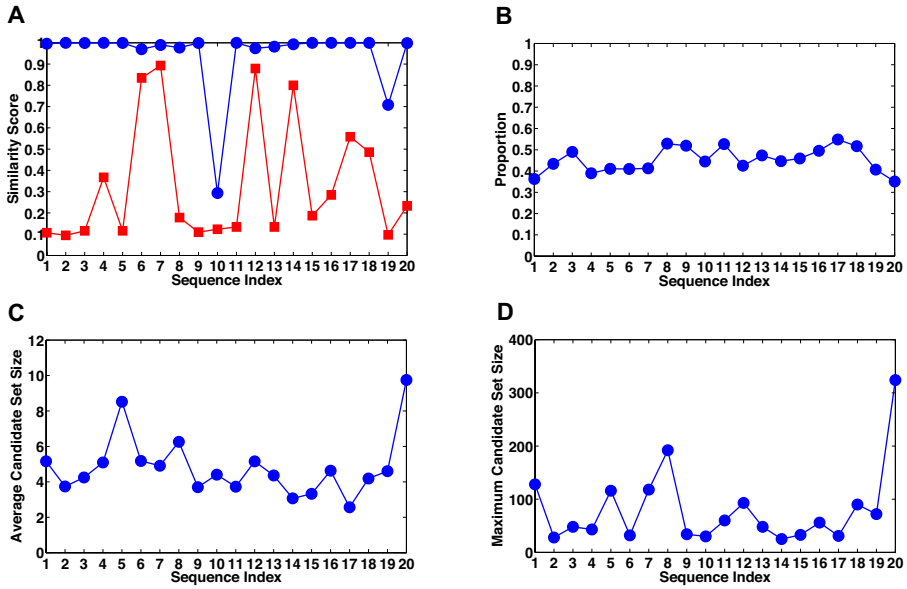


Figure 1.7: Performance of the method for random sequences. 20 randomly generated 5000-mers are sequenced using 7-mer probes assuming 5% positive and 5% negative random errors in the probe spectrum. Each target is replicated 8 times and separated into fragments of length randomly within $[180, 220]$. (A) Similarity score comparison between the proposed method (blue circles) and a control method (red squares). The control method does not generate complete sets of possible sequences for fragments, i.e. each candidate solution set contains exactly one candidate. (B) In contrast, due to reconstruction degeneracy and biochemical errors, the proportion of correctly reconstructed fragments is less than one half on average. (C), (D) The average and maximum number of candidates of fragments used in the assembly. The proposed method significantly outperforms the control method and allows to uniquely choose one out of up to 300 degenerate solutions of a fragment.

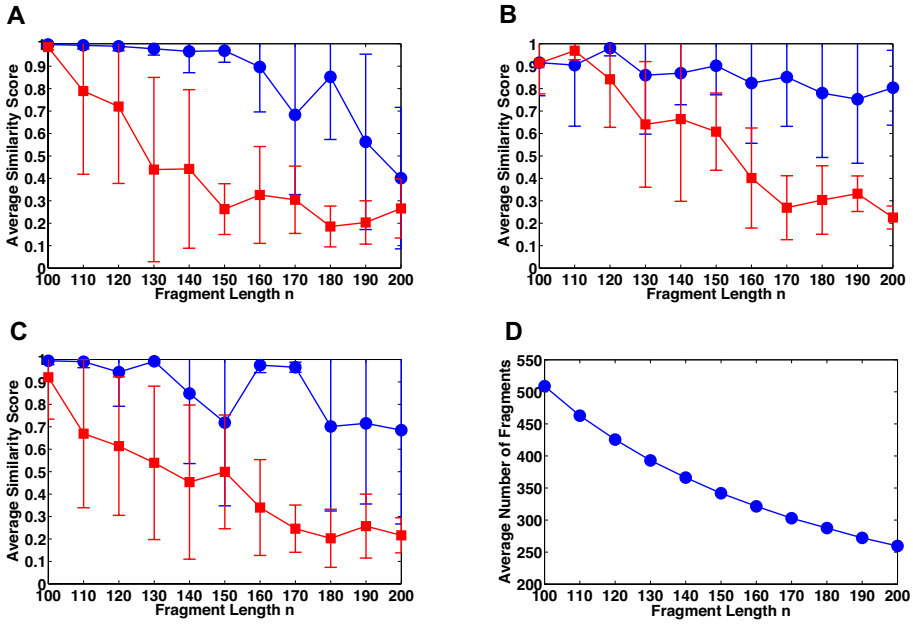


Figure 1.8: Performance of the method on natural sequences. The average similarity score over ten independent simulation runs on the 5000-mer prefix of three natural sequences (GenBank accession numbers JA638618 (A), AEQT01000438 (B) and AFZZ01000001 (C)), as a function of the fragment length n used to separate replicas. The results of both the proposed method (blue circles) and a control method (red squares) without generating a complete set of possible sequences for each fragment are shown. Error bars represent the standard deviation. Each 5000-mer is replicated 10 times and sequenced with 7-mers. 5% positive and 5% negative errors are included in the spectrum of each fragment. (D) shows the average number of fragments need to be sequenced as a function of the fragment length n.

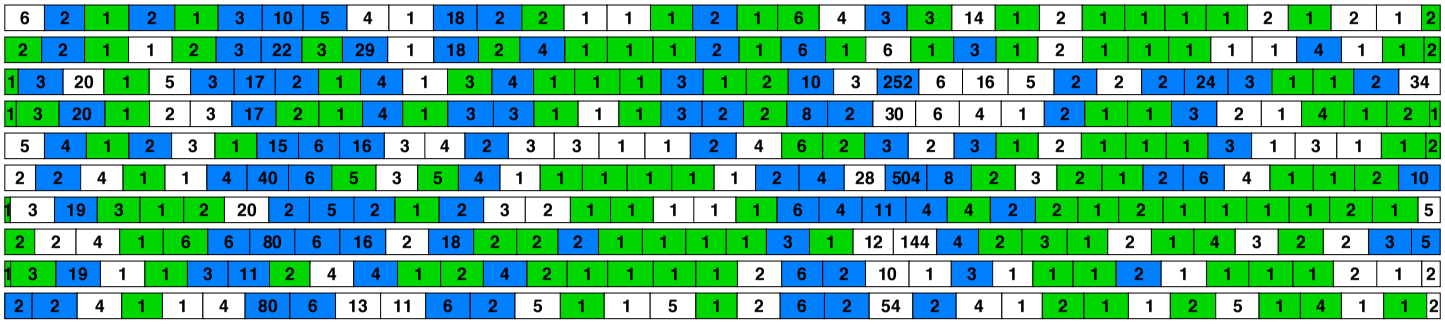


Figure 1.9: Visualization of replicas of the 5000-mer prefix of sequence JA638618 in one simulation attempt. Each row represents one replica of the sequence. Green bars represent fragments that are correctly reconstructed before solution enumeration, while blue bars represent fragments that are not correctly reconstructed, but the correct solutions appear in their candidate solution sets. The number of consistent solutions are shown in each fragment.

We keep moving forward, opening new doors, and doing new things, because we're curious and curiosity keeps leading us down new paths.

Walt Disney

2

Self-Assembly Optimization Method Studies Using GPU Computing

Self-assembly[100,101,102] is a process where simple building blocks spontaneously assemble into structures of higher complexity. In this chapter, in light of examples from colloidal engineering where particles interact in short range[69], I explore two self-assembly optimization methods with GPU computing[75,79,80,24]: controlling the short-ranged

interactions between particles[50], and controlling the concentration of different types of particles in the system[70]. The first method is simulated in canonical ensemble, and the second method in both canonical and grand canonical ensembles. Simulation results are consistent with theoretical predictions in both cases. When optimizing concentrations, the nonequilibrium behavior of the particle system is also unveiled thanks to the enormous computation capacity of GPUs.

The structure of the chapter is as follows. Section 2.1 presents the physical model of the simulation, including equations of forces, the Verlet integration algorithm, and the valence and crosstalk model. Section 2.2 gives an overview of the GPU computing technique, and elaborates two parallelization methods applied in the simulation. Section 2.3 discusses the calibration of the particle system. Section Section 2.4 and Section 2.5 present results of applying GPU computing to simulate two self-assembly optimization methods. Particularly, Section 2.4 discusses the method of controlling the interactions between particles, while Section 2.5 presents the method of controlling the concentrations of different types particles. Section 2.6 summarizes the chapter.

2.1 The Physical Model

The physical model of the simulation is inspired by an example in colloid engineering[69], where a particle system is composed of colloid and solvent particles, and the interaction between particles are short-ranged. If colloid particles are not charged, their Brownian motion does not depend on the molecular details of the solvent, but only on its temperature, density and viscosity. Therefore, we use Dissipative Particle Dynamics (DPD), a method usually used to simulate the dynamics of simple and complex fluids[49,40], to simulate the particle interactions. DPD is a coarse-grained Molecular Dynamics (MD) method, where solvent particles represent clusters of solvent molecules and interact with others through pair-wise forces. DPD resembles the Brownian Dynamics (BD) method. The main difference is that in BD, frictional and random forces do not conserve momentum, whereas in DPD they do. Figure 2.1A illustrates a schema of the DPD method, and Figure 2.1B shows a snapshot of our simulation where the system is composed of 6 identical colloid particles and $\sim$ 400 solvent particles.

2.1.1 Newton motion

In DPD, the evolution of interacting particles is governed by Newton's equations of motion

$$\frac{\mathrm{d}\mathrm{r}_i}{\mathrm{d}t} = \mathrm{v}_i, \qquad m\frac{\mathrm{d}\mathrm{v}_i}{\mathrm{d}t} = \mathrm{f}_i, \tag{2.1}$$

where particle i could be either colloid or solvent particle. To simplify the model, the mass of all colloid and solvent particles are set equal ($= m$) in the simulation. The force

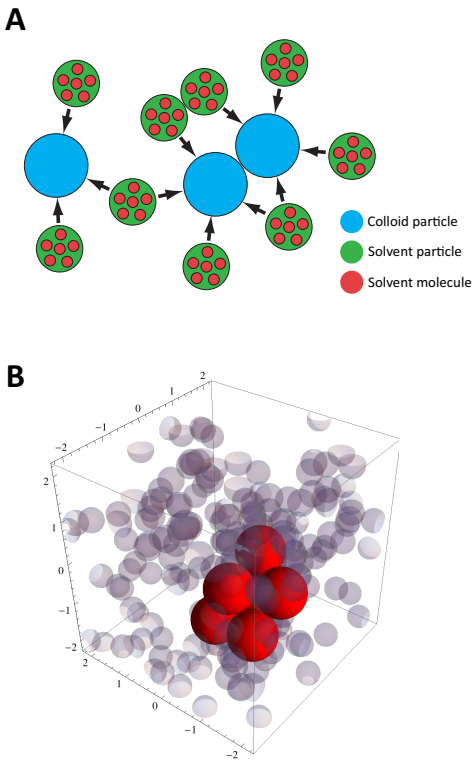


Figure 2.1: A illustrates the Dissipative Particle Dynamics (DPD). DPD is a coarse-grained Molecular Dynamics (MD) method, where solvent particles (green) represent clusters of solvent molecules (red) and interact with others through pair-wise forces. In the schema, colloid particles (blue) assembles as a result of the random kicks of the surrounding solvent particles. Note that colloid particles could also be clusters of some smaller elements. **B** is a snapshot of the actual DPD simulation, where the system contains 6 identical colloid particles (red) and $\sim$ 400 solvent particles (transparent gray).

exerted on particle i is composed of three parts:

$$\mathrm{f}_i = \sum_{d_C} \mathrm{f}_C + \sum_{d_D} \mathrm{f}_D + \sum_{d_R} \mathrm{f}_R, \tag{2.2}$$

where f_C, f_D and f_R are conservative force, dissipative force and random force respectively, and the summations are over particles inside cutoff distances, d_C, d_D and d_R respectively, centered by particle i. Since each particle is surrounded by both colloid and solvent particles, depending on the identity of particle i (colloid or solvent), the above formula can be expanded to

$$\mathrm{f}_i^c = \sum_{d_C^{cc}} \mathrm{f}_C^{cc} + \sum_{d_C^{cs}} \mathrm{f}_C^{cs} + \sum_{d_D^{cc}} \mathrm{f}_D^{cc} + \sum_{d_D^{cs}} \mathrm{f}_D^{cs} + \sum_{d_R^{cc}} \mathrm{f}_R^{cc} + \sum_{d_R^{cs}} \mathrm{f}_R^{cs} \tag{2.3}$$

and

$$\mathrm{f}_i^s = \sum_{d_C^{ss}} \mathrm{f}_C^{ss} + \sum_{d_C^{cs}} \mathrm{f}_C^{cs} + \sum_{d_D^{ss}} \mathrm{f}_D^{ss} + \sum_{d_D^{cs}} \mathrm{f}_D^{cs} + \sum_{d_R^{ss}} \mathrm{f}_R^{ss} + \sum_{d_R^{cs}} \mathrm{f}_R^{cs}, \tag{2.4}$$

where c and s superscripts represent colloid and solvent respectively.

2.1.2 Conservative force

When two colloid particles are identical, we use a steep 96-48 Lennard-Jones potential[55] to model the binding energy V_{ij} between them:

$$V_{ij} = \begin{cases} \varepsilon_a \left[\left(\dfrac{r_m}{r_{ij}} \right)^{96} - 2 \left(\dfrac{r_m}{r_{ij}} \right)^{48} \right] & \text{if } r_{ij} < d_C^{cc} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where $r_{ij} = |\mathrm{r}_{ij}| = |\mathrm{r}_i - \mathrm{r}_j|$ is the distance between centers of particle i and j, r_m is the distance between two particle centers at which the potential reaches its minimum,

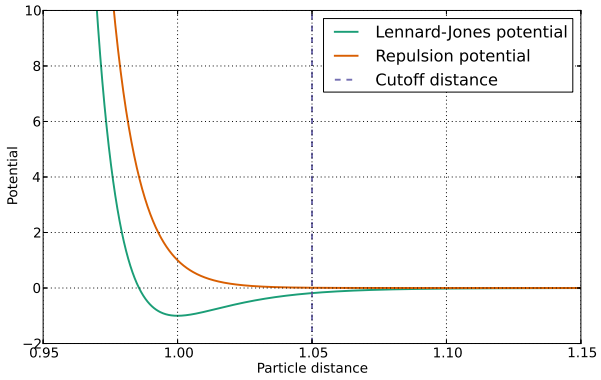


Figure 2.2: The potential energies between colloid particles. When two colloid particles are identical, or coated with attracted DNA strands, we use the Lennard-Jones potential (green curve) to model their binding energy. The distance r_m where the potential reaches its minimum is set to 1, and the depth of the potential well $\varepsilon_a = 1$. When the colloid particles are coated with repelled DNA strands, we use the repulsion part (red curve) of the Lennard-Jones potential to model the energy. The repelling strength ε_r is set to 1 in the plot. The purple dashed line shows the cutoff distance $d_C^{cc} = 1.05$ used in the simulation.

and ε_a is the depth of the potential well, or the attracting strength. The green curve in Figure 2.2 shows the Lennard-Jones potential. In the simulation, without loss of generality, we set $r_m = 1$. The conservative force between two identical colloid particles is then calculated by taking the first derivative of the potential:

$$\mathrm{f}_{C,ij}^{cc} = -\frac{\mathrm{d}V_{ij}}{\mathrm{dr}_{ij}} = \begin{cases} \dfrac{96\varepsilon_a}{r_{ij}^{49}}\left(\dfrac{1}{r_{ij}^{48}} - 1\right)\mathrm{e}_{ij} & \text{if } r_{ij} < d_C^{cc} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where $\mathrm{e}_{ij} = \mathrm{r}_{ij}/r_{ij}$ is the unit vector pointing from j to i.

Technology is rapidly evolving to allow *colored* colloid particles, e.g. particles coated with DNA strands[16], in the self-assembly so that their interactions are highly controllable. When two colloid particles are coated with attracting DNA strands, we use the same potential (2.5) and force (2.6) to model their interaction. On the other hand,

when two colloid particles are coated with repelled DNA strands, we use the repulsion part of (2.5) to model their potential energy:

$$V_{ij} = \begin{cases} \varepsilon_r \left(\frac{r_m}{r_{ij}} \right)^{96} & \text{if } r_{ij} < d_C^{cc} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

where ε_r is the repelling strength. The conservative force is then calculated as

$$\mathrm{f}_{C,ij}^{cc} = -\frac{\mathrm{d}V_{ij}}{\mathrm{d}r_{ij}} = \begin{cases} \frac{96\varepsilon_r}{r_{ij}^{97}} \mathrm{e}_{ij} & \text{if } r_{ij} < d_C^{cc} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

The colloid-solvent and solvent-solvent conservative forces are simple elastic force:

$$\mathrm{f}_{C,ij}^{cs} = \begin{cases} k^{cs} \left(1 - \frac{r_{ij}}{d_C^{cs}} \right) \mathrm{e}_{ij} & \text{if } r_{ij} < d_C^{cs} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

$$\mathrm{f}_{C,ij}^{ss} = \begin{cases} k^{ss} \left(1 - \frac{r_{ij}}{d_C^{ss}} \right) \mathrm{e}_{ij} & \text{if } r_{ij} < d_C^{ss} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

where k^{cs} and k^{ss} are the elastic constants.

2.1.3 Dissipative and random forces

Dissipative and random forces together forms the thermostat of DPD. Dissipative force is the frictional force due to viscous resistance within the fluid. It reduces the relative velocity of particle pairs. On the other hand, random force represents the stochastic part of the dynamics. It compensates the eliminated degree of freedom due

to coarse-graining.

In the simulation, we set the range of dissipative and random force equal, denoted as $d_{DR} = d_D = d_R$. The dissipative and random forces between two colloid particles are

$$\mathrm{f}^{cc}_{D,ij} = \begin{cases} -\gamma\omega_{D,ij}(\mathrm{v}_{ij}\bullet\mathrm{e}_{ij})\mathrm{e}_{ij} & \text{if } r_{ij} < d^{cc}_{DR} \\ \mathrm{o} & \text{otherwise} \end{cases} \tag{2.11}$$

and

$$\mathrm{f}^{cc}_{R,ij} = \begin{cases} \frac{1}{\sqrt{\Delta t}}\sigma\omega_{R,ij}\xi_{ij}\mathrm{e}_{ij} & \text{if } r_{ij} < d^{cc}_{DR} \\ \mathrm{o} & \text{otherwise} \end{cases} \tag{2.12}$$

In the above formulas, Δt is the integraion time step, and ξ_{ij} is a diagonal 3-by-3 matrix whose diagonal elements are independent random numbers uniformly distributed in $(-\sqrt{3}, \sqrt{3})$. The magnitude relation between the dissipative and random forces is determined by the Fluctuation Dissipation Theorem[77]:

$$\sigma^2 = 2\gamma k_B T, \quad \omega_{D,ij} = (\omega_{R,ij})^2, \tag{2.13}$$

where k_B is the Boltzman constant, T is the absolute temperature. The *kinetic temperature* $k_B T$ can be calculated from the mean kinetic energy of all particles in the system:

$$\frac{3}{2}k_B T = \frac{1}{2}m\overline{v^2}. \tag{2.14}$$

In the rest of this chapter, we use *temperature* to represent the *kinetic temperature* for simplicity. $\omega_{R,ij}$ is a linear function of the distance between particle i and j:

$$\omega_{R,ij} = 1 - \frac{r_{ij}}{d_{DR}}. \tag{2.15}$$

The colloid-solvent and solvent-solvent dissipative and random forces have the same forms, except switching d^{cc}_{DR} to d^{cs}_{DR}, d^{ss}_{DR}, and the integration time step Δt to a larger time step ΔT (more on this in the next section).

2.1.4 Verlet integration

Verlet integration[98] is a numerical method used to integrate Newton's equations of motion. It is frequently used to calculate trajectories of particles in molecular dynamics simulations and computer graphics. To integrate our system of colloid and solvent particles forward, we use a variant of the Velocity Verlet algorithm[94]. In particular, to simulate the detailed motions of colloid particles while not significantly increase the computation complexity due to the large amount of solvent particles, we introduce two integration time steps: a small time step Δt for colloid and a large time step $\Delta T \gg \Delta t$ for solvent. The integration steps of colloid is the standard Velocity Verlet algorithm:

$$
\begin{aligned}
&1.\quad \mathrm{r}_c(t+\Delta t) = \mathrm{r}_c(t) + \mathrm{v}_c(t)\Delta t + \frac{1}{2}\mathrm{a}_c(t)\Delta t^2;\\
&2.\quad \mathrm{v}_c(t+\frac{1}{2}\Delta t) = \mathrm{v}_c(t) + \frac{1}{2}\mathrm{a}_c(t)\Delta t;\\
&3.\quad \text{Derive } \mathrm{a}_c(t+\Delta t) \text{ using } \mathrm{r}_c(t+\Delta t) \text{ and } \mathrm{v}_c(t+\frac{1}{2}\Delta t);\\
&4.\quad \mathrm{v}_c(t+\Delta t) = \mathrm{v}_c(t+\frac{1}{2}\Delta t) + \frac{1}{2}\mathrm{a}_c(t+\Delta t)\Delta t.
\end{aligned}
\tag{2.16}
$$

In the above steps, r_c, v_c and a_c respectively represent positions, velocities and accelerations of colloid particles. When deriving a_c, we only use forces between colloid particles, i.e. f^{cc}_C, f^{cc}_D and f^{cc}_R.

The complete integration steps of both colloid and solvent particles are:

$$
\begin{aligned}
&1.\quad \mathrm{v}_{all}(t+\tfrac{1}{2}\Delta T) = \mathrm{v}_{all}(t) + \tfrac{1}{2}\mathrm{a}_{all}(t)\Delta T;\\
&2.\quad \text{Iterate the above colloid integration cycle (2.16) } \Delta T/\Delta t \text{ times;}\\
&3.\quad \mathrm{r}_{s}(t+\Delta T) = \mathrm{r}_{s}(t) + \mathrm{v}_{s}(t)\Delta T;\\
&4.\quad \text{Derive } \mathrm{a}_{all}(t+\Delta T) \text{ using } \mathrm{r}_{all}(t+\Delta T) \text{ and } \mathrm{v}'_{all}(t+\tfrac{1}{2}\Delta T);\\
&5.\quad \mathrm{v}_{all}(t+\Delta T) = \mathrm{v}'_{all}(t+\tfrac{1}{2}\Delta T) + \tfrac{1}{2}\mathrm{a}_{all}(t+\Delta T)\Delta T. \qquad (2.17)
\end{aligned}
$$

r_{all}, v_{all} and a_{all} are positions, velocities and accelerations of all particles respectively, including colloid and solvent particles. When deriving a_{all}, we use forces f^{cs}_{C}, f^{ss}_{C}, f^{cs}_{D}, f^{ss}_{D} and f^{cs}_{R}, f^{ss}_{R}, excluding f^{cc}_{C}, f^{cc}_{D} and f^{cc}_{R}. $\mathrm{v}'_{all}(t+\frac{1}{2}\Delta T)$ in step 4 and 5 is different from $\mathrm{v}_{all}(t+\frac{1}{2}\Delta T)$ in step 1 in that the velocities of colloid particles have been changed in step 2.

To summarize, the above integration algorithm makes solvent particles move in a large time step ΔT, while features a 'slow motion' of colloid particles with a smaller time step Δt, i.e. step 2 in (2.17). This effectively simulates the interactions between all particles in the system, while prevents the simulation from being computationally complex due to large number of solvent particles. In the simulation, we set $\Delta T/\Delta t = 70$.

2.1.5 Other settings

While most parameters in the simulation are user-defined, some parameters are inferred from others to ensure the physical correctness of the system.

Particle radiuses

The radius of colloid particle is set so that when two colloid particles i and j touch each other exactly, the Lennard-Jones potential reaches minimum, or $r_{ij} = r_m = 1$. Therefore, the radius of colloid particle is

$$r_c = \frac{r_m}{2} = 0.5. \tag{2.18}$$

The radius of solvent particle is set so that when two colloid particles are closed and start exerting dissipative and random forces on each other, one solvent particle can still fit between them. Therefore, the radius of solvent particle is

$$r_s = \frac{d_{DR}^{cc} - 2r_c}{2}. \tag{2.19}$$

Force ranges

Depending on particle and force types, interactions happen either when particles are sufficiently closed, or only when they touch each other. The cutoff distances of conservative forces are set as follows:

$$d_C^{cc} = 2.1r_c, \quad d_C^{cs} = r_c + r_s, \quad d_C^{ss} = 2r_s. \tag{2.20}$$

The cutoff distances of dissipative and random forces are

$$d_{DR}^{cc} = 3r_c, \quad d_{DR}^{cs} = 0.5d_{DR}^{cc} + r_s, \quad d_C^{ss} = 2r_s. \tag{2.21}$$

Box size and number of solvent particles

In the simulation, all particles are inside a cubic *box*. The number of colloid particles n_c and the *volume fraction* φ are user defined. The volume of the box V is calculated as

$$V = \frac{\frac{4}{3}\pi r_c^3 n_c}{\varphi}. \tag{2.22}$$

The side length of the box is then $b = V^{1/3}$. We also predefine the solvent density as ϱ. Then the number of solvent particles is

$$n_s = \frac{V\varrho}{m}. \tag{2.23}$$

Boundary condition

To prevent the boundary of the box from interfering the self-assembly, we use *periodic boundary condition* in the simulation. Assuming the center of the box is the origin, when any of the three coordinates of particle i, r_{ix}, r_{iy} or r_{iz}, is outside the box, it is added a multiple of the box side length b to set it back into the box:

$$\text{If } |r_{i\{x,y,z\}}| > \frac{b}{2}, \quad r_{i\{x,y,z\}} \leftarrow r_{i\{x,y,z\}} + kb \ (k \in \mathbb{Z}) \text{ s.t. } |r_{i\{x,y,z\}}| \leq \frac{b}{2}. \tag{2.24}$$

2.1.6 Canonical and grand canonical ensembles

In statistical mechanics, a canonical ensemble is the statistical ensemble that is used to represent the possible states of a mechanical system of particles, which is in thermal equilibrium with a heat bath[38]. The system can only exchange energy with the heat bath, while its composition, volume and shape are kept constant. To simulate a such

system, we can simply keep the number of particles unchanged, and the heat transfer is fulfilled through dissipative force.

A grand canonical ensemble is the statistical ensemble that is used to represent the possible states of a mechanical system of particles maintained in both thermal and chemical equilibrium with a reservoir[38]. The system can exchange both energy and particles with the reservoir, so that various possible states of the system can differ in both their total energy and total number of particles, while the volume and shape of the system are kept unchanged in all possible states. Computation on grand canonical ensemble is highly expensive as the reservoir is assumed to be significantly larger than the system itself and therefore requires the coexistence of huge amount of particles in the simulation. To approximate the grand canonical ensemble, for every t_{add} time steps, we add new colloid particles into the system to keep the concentrations of different types of *monomers*, or colloid particles without binding, roughly constant, while removing a fraction of all colloid particles from the system every t_{remove} time steps to prevent the total mass of the system from growing. To ensure the removing is uniformly across all colloid particles, we implement a Breath-First Search (BFS) algorithm[62] to identify all assembled structures and monomers currently in the system, and remove a fraction of them from the system according to a constant probability p_{remove}.

2.1.7 Valence and crosstalking interaction

Except for the binding preference induced by DNA coating, other local binding laws, such as *binding sites*[56] and *valence*, could also exist. Valence is the maximum number of bonds a particle can form. If two particles with attracted colors come closed but any of them already reaches its valence, they will repel each other. In addition to strict attraction and repulsion, low-energy non-specific bindings[51], or *crosstalking interactions*,

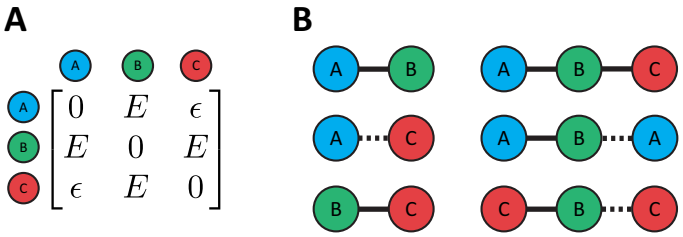


Figure 2.3: A simple example of the valence and crosstalk model. There are three types of particles. A (blue) and C (red) has valence 1, while B (green) has valence 2. **A** shows the adjacency matrix. Strong bonds with energy E can form between A, B and B, C, while weak bond with energy $\varepsilon \ll E$ can form between A and C due to crosstalk. Particles of the same type repel each other. **B** shows all structures that can be formed with these three particle types. Solid lines represent strong bonds, and dashed lines are weak bonds. In our model, when B binds to two particles of the same type, the second bond is weak.

ubiquitously exist between self-assembly components in both natural and synthetic systems. For example, weak bonds inevitably form between different types of proteins due to protein structure constraints. As a result, an exponential number of undesired structures could form in the system, which poses a challenge of assembling a specific structure.

In the simulation of controlling particle concentrations to optimize yield, we use a model where particles have both valence and crosstalking interactions. Figure 2.3 shows a simple example of a such model. There are three types of particles, denoted as A (blue), B (green) and C (red). A and C has valence 1, while B has valence 2. Figure 2.3A shows the *adjacency matrix*. Strong bonds with energy E can form between A, B and B, C, while weak bond with energy $\varepsilon \ll E$ can form between type A and C due to crosstalk. Particles of the same type repel each other. Figure 2.3B enumerates all structures that can be formed in this model. Note that when B binds to two particles of the same type, the second bond is weak.

2.2 GPU Computing

Graphics Processing Unit (GPU) is a processor dedicated to 3D graphics rendering through its specialized multiprocessor architecture. It have been grown to become extremely powerful and significantly exceed Central Processing Unit (CPU) in raw computing power. In the 1999–2000 timeframe, scientists from various fields started using GPUs to accelerate a range of scientific applications, subject to the constraints of programming through graphics APIs. In November 2006, NVIDIA introduced the CUDA architecture[74,89], which includes new components designed strictly for GPU computing and aimed to alleviate many of the limitations that prevented previous graphics processors from being legitimately useful for general-purpose computing. It makes GPU fully programmable, and offers seamless experience for developers with familiar languages such as C, C++, and Fortran. Today, GPU computing momentum is growing faster than ever before. It has been applied to computational fluid dynamics[25,26], medical imaging[93,91], molecular dynamics[92,8] and many other fields. Users achieves unprecedented performance by switching to GPUs, over 100x speedup compared to CPUs in some cases. All simulation results in this chapter are produced on a small server with three GPUs: two NVIDIA Tesla C2075 and one NVIDIA Quadro 6000.

The rest of this section presents two examples on how we use GPU parallelization to achieve fast DPD simulation. The first example elaborates how we efficiently perform computation over an ensemble of independent particle systems. The second example presents a simple GPU-specific solution for each particle to only iterate through its surrounding particles within a short range when computing the forces.

2.2.1 Computation on particle system ensemble

To statistically study the self-assembly of colloid, we sometimes need to collect data from an ensemble of independent particle systems, or *configurations*. More specifically, in the simulation, we have many particle systems started from different and independent random states, where particles are assigned uniformly random positions and normally random velocities. We let all systems run for the same number of time steps, then snapshoot and collect their final states. This also resembles the experiment[69], where colloid particles are placed in independent microwells to self-assemble. With data from many independent particle systems, we can statistically study e.g. under a certain temperature, what is the probability that the colloid particles assemble into a specific ground state structure. Figure 2.4 shows a schema of the strategy.

Notice that there are two layers of parallelization in the computation. On a large scale, all configurations are independent and *embarassingly parallel*. On a small scale, in the Verlet integration process of each system, each particle independently iterates through its surrounding particles to calculate its resultant force, and moves forward using its own position, velocity and force information. We parallelize both of them to leverage the massive parallel architecture of GPU.

Our parallel force calculation strategy is illustrated in Figure 2.5. Particle data (positions, velocities and forces) are stored in consecutive memory. Particularly, colloid and solvent data are separated into two parts of the memory (the blue and orange bar). Since $\Delta t \ll \Delta T$, the integration is in the *colloid cycle* (2.16) most of the time, where each colloid particle iterates through other colloid particles in the same configuration to calculate its force (step 3 of (2.16)). This is illustrated as the black arrow in the figure. When the simulation reaches step 4 of the *solvent cycle* (2.17), as the red arrows show,

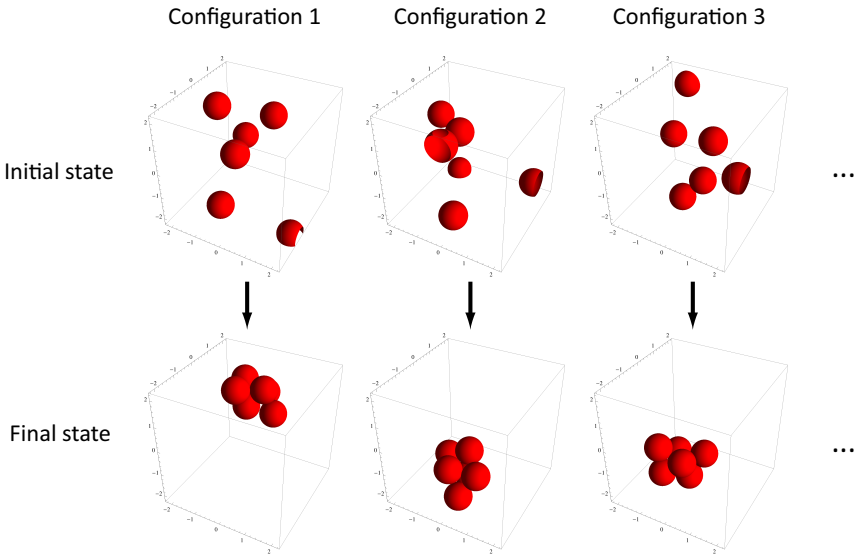


Figure 2.4: Schema of computation over independent particle systems. Red spheres represent colloid particles. Solvent particles are not shown for visual purpose. In the simulation, we have many particle systems, or configurations, started from different and independent random states (first row), where particles are assigned uniformly random positions and normally random velocities. We let all systems run for the same number of time steps, then snapshoot and collect their final states (second row). With data from many independent particle systems, we can statistically study e.g. under a certain temperature, what is the probability that the colloid particles assemble into a specific ground state structure.

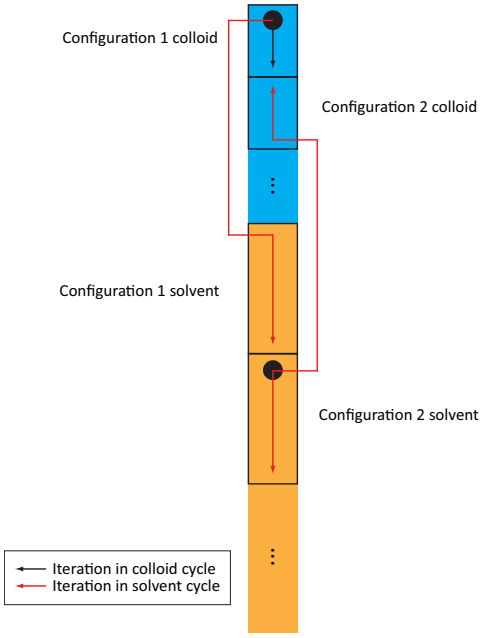


Figure 2.5: Parallel force calculation for particle system ensemble. Particle data (positions, velocities and forces) are stored in consecutive memory (the vertical bar). More specifically, colloid (blue) and solvent (orange) data are separated into two parts. The simulation is in the *colloid cycle* (2.16) most of the time, where each colloid particle iterates through other colloid particles in the same configuration to calculate its force (black arrow). When the simulation reaches step 4 of the *solvent cycle* (2.17), to calculate the forces, each colloid particle needs to iterate through solvent particles in the same configuration, while each solvent particle should iterate though both colloid and solvent particles in the same configuration (red arrows).

each colloid particle needs to iterate through solvent particles in the same configuration to calculate its force, while each solvent particle should iterate though both colloid and solvent particles in the same configuration. In other steps of the integration, particles are completely parallel from each other – they simply use their own position, velocity and force information to move forward.

2.2.2 Spatial subdivision

If each particle must examine all other particles to calculate the forces, the time complexity of a serial DPD simulation is $\mathcal{O}(n^2)$, and $\mathcal{O}(n)$ for a parallel DPD, where n is the total number of particles in a system. Since all forces in our simulation are short-ranged, we can use a *spatial subdivision* strategy to further reduce the time complexity to $\mathcal{O}(\hat{k})$, where $\hat{k}$ is the average number of surrounding particles of each particle. When the volume fraction φ is low so that $\hat{k} \ll n$, applying spatial subdivision can achieve significant speedup.

Figure 2.6 illustrates our spatial subdivision strategy in 2D. We use a uniform grid[34] to subdivide the box into a grid of equal-side cells. The side length of the cell is equal to the longest force range, which is d_{DR}^{cc} in our case. In each time step, we determine the cell each particle is located. Then, when calculating the forces, each particle only need to examine particles in its own and neighboring cells to determine which particles are inside the force ranges. The number of cells each particle needs to examine is $3 \times 3 = 9$ in 2D and $3 \times 3 \times 3 = 27$ in 3D. For example, when calculating the dissipative and random forces exerted on particle 3 in the figure, we notice that it is in cell 5, therefore, we should iterate through particles in cell 0, 4, 8, 1, 5, 9, 2, 6 and 10 to determine if there is any particle within the cutoff distance d_{DR}^{cc} from particle 3 (particle 4 and 5 inside the red dashed circle in the example). Note that the algorithm should take the periodic boundary condition into account. For example, particles in cell 3 should examine particles in cell 2, 6, 3, 7, as well as cell 0, 4, 12, 14, 15.

To implement this algorithm on GPU, a simple approach is to take advantage of the *atomic operations*[76], supported on GPUs with *compute capability* greater than or equal to 1.1. Atomic operations allow multiple GPU threads to update the same value

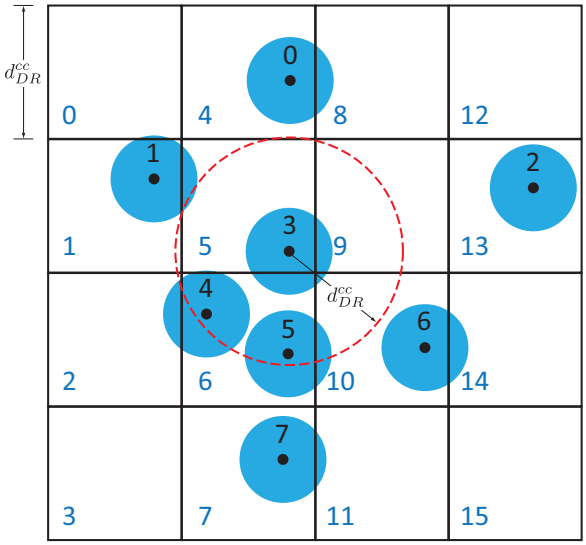


Figure 2.6: The spatial subdivision strategy in 2D. We subdivide the box into a grid of equal-side cells, whose side length is equal to the longest force range, d_{DR}^{cc} in our case. When calculating the forces, each particle only need to examine particles in its own and neighboring cells to determine which particles are inside the force ranges. For example, when calculating the dissipative and random forces exerted on particle 3 in the figure, particles in cell 0, 4, 8, 1, 5, 9, 2, 6 and 10 should be examined through, and eventually particle 4 and 5 within the cutoff distance d_{DR}^{cc} (red dashed circle) will be found.

in *global memory* simultaneously without conflicts. As Table 2.1 shows, we use two containers in the implementation: a *counter* array (2nd column) to store the number of particles in each cell so far, and an *index* array (3rd column) to store the particle indices of each cell. Note that when two particles happen to penetrate each other, the steep Lennard-Jones potential will force them apart. Therefore, we can empirically determine the maximum number of particles in each cell and preallocate sufficient memory for the index array. Both containers are updated in each time step. Specifically, the counter array is initialized to zero at the start of each time step. The simulation runs with one thread per particle. Each particle calculated which cell it is in, and uses the `atomicAdd` function[76] to atomically increment the counter of this cell. It then writes its index into the index array at the corresponding position using a scattered global write.

There are other more efficient methods to perform spatial subdivision. We choose this approach mainly for its simplicity. Admittedly, the atomic operation is considered relatively expensive in GPU computing, as when multiple threads attempt to write to the same memory unit, the operations will be serialized. Therefore, we only use this algorithm when there are large number of particles and the volume fraction φ is low. When the volume fraction is high, the speedup gain from applying spatial subdivision is comparable to the extra time in refreshing the counter and index arrays. In this case, we choose to iterate though all particles when calculating the forces.

Cell index	Particle Count	Particle index
0	0	
1	1	1
2	0	
3	0	
4	1	0
5	1	3
6	2	4, 5
7	1	7
8	0	
9	0	
10	1	6
11	0	
12	0	
13	1	2
14	0	
15	0	

Table 2.1: In the implementation of spatial subdivision, we use two containers: a *counter* array (2nd column) to store the number of particles in each cell so far, and an *index* array (3rd column) to store the particle indices of each cell. Numbers in this table come from Figure 2.6. The counter array is initialized to zero at the start of each time step. The simulation runs with one thread per particle. Each thread computes which cell its particle is in, and uses the `atomicAdd` function [76] to atomically increment the counter of this cell. It then writes its index into the index array at the corresponding position using a scattered global write.

2.3 Calibration

Before proceeding to simulate various self-assembly optmization methods, it is important to verify that our GPU program produces physically correct results. This section presents various data collected through both canonical and grand canonical ensemble simulations to prove the physical correctness of our program.

2.3.1 Canonical ensemble

To set the particle system to a specified temperature $k_B T$, we randomly initialize the velocities of all particles (colloid and solvent) according to standard normal distribution, and rescale these velocities so that

$$\frac{1}{2} m \overline{v^2} = \frac{3}{2} k_B T, \tag{2.25}$$

where $\frac{1}{2} m \overline{v^2}$ is the mean kinetic energy of all particles. Initial positions of particles are uniformly random, subject to the constraint that colloid particles do not penetrate each other. Since we do not have constraints on the initial positions of solvent particles, it is possible that some solvent particles penetrate other particles at the starting, which results in strong repulsions and accelerations. Therefore, the temperature of the system is expected to increase drastically when the simulation starts. On the other hand, what we should confirm is that after this initial temperature pulse, whether the system is capable to cool down itself back to the predefined temperature, instead of keep diverging.

To verify this, we set up a canonical ensemble particle system with 100 identical colloid particles. The volume fraction is set to $\varphi = 0.02$[*] and the input temperature

[*] In the rest of this chapter, we use $\varphi = 0.02$ unless otherwise indicated.

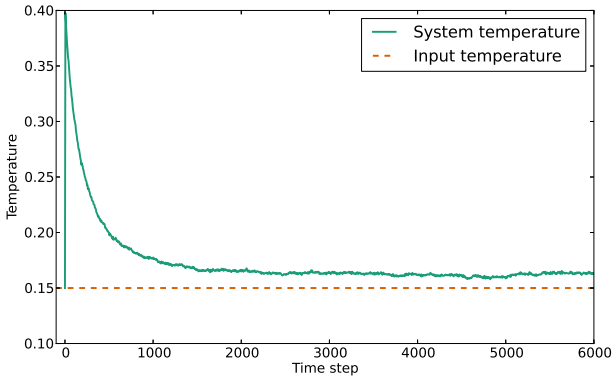


Figure 2.7: The system temperature of a canonical ensemble particle system as a function of simulation time. There are 100 identical, or uncolored, colloid particles in the system. The volume fraction φ is set to 0.02, and the number of solvent particles is 7853 calculated from (2.22) and (2.23). The input temperature is set to 0.15 (red dashed line). After the initial temperature pulse, the system cools down towards the input temperature instead of keep diverging.

is 0.15. Figure 2.7 shows the system temperature as a function of the simulation time. It shows that after the initial temperature pulse, the system cools down towards the input temperature. Because of the finite integration time step, the system temperature never goes back exactly to the input temperature, but converges to a value ($\sim$ 0.164) acceptably higher than the input.

We also need to confirm whether the bond breaking time is consistent with thermodynamics. In thermodynamics, under a certain temperature $k_B T$, the probability that a system jumps out from an energy well of depth ε_a, or the probability for the bond between two colloid particles to break is

$$p_{break} \sim \exp\left(-\frac{\varepsilon_a}{k_B T}\right). \tag{2.26}$$

Since the bond breaking frequency is

$$f_{break} = \frac{1}{t_{break}} \sim p_{break}, \tag{2.27}$$

where t_{break} is the bond breaking time, we have

$$\ln\left(\frac{1}{t_{break}}\right) \sim -\varepsilon_a \left(\frac{1}{k_B T}\right), \tag{2.28}$$

or $\ln\left(1/t_{break}\right)$ is a linear function of $1/k_B T$ with slope $-\varepsilon_a$. To verify this, we initialize a system with 100 dimer (two-particle structure) and set the depth of the potential well to $\varepsilon_a = 1$. We let the system run for 10^4 time steps and monitor the time these dimers break at various temperature. Figure 2.8A shows the average dimer breaking time as a function of the temperature. Notice that when the temperature is smaller than ~ 0.18, the bond breaking time increases drastically. The curve becomes flat again at low temperature as the 10^4 time steps is not sufficient for the bond to break at those temperatures. Figure 2.8B aims to confirm (2.28). The red dashed line has slope -1. Except for the low temperature points, data from the simulation is consistent with (2.28).

2.3.2 Grand canonical ensemble

We also examine the physical properties of the grand canonical ensemble simulation. We use a valence and crosstalk model composed of three types of particles A, B and C,

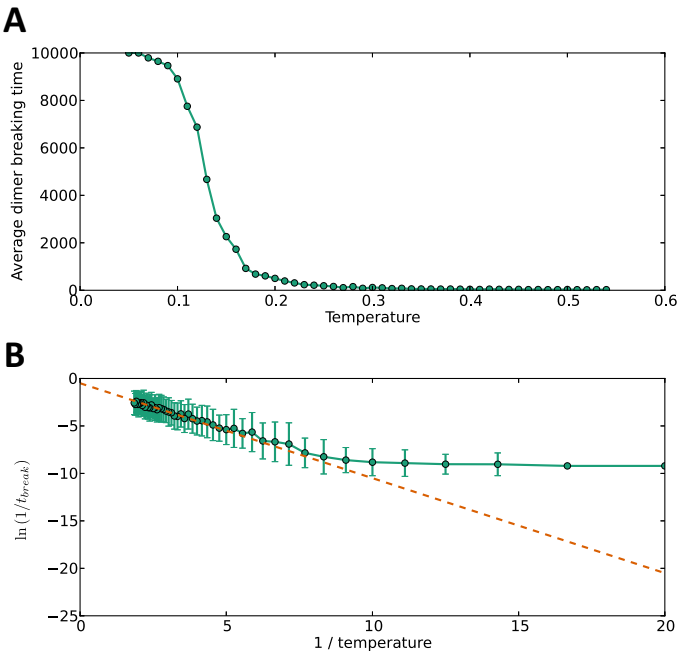


Figure 2.8: We initialize a system with 100 dimer (two-particle structure) and set $\varepsilon_a = 1$. The system runs for 10^4 time steps under various temperatures. **A** shows the average dimer breaking time as a function of the temperature. **B** plots the average $\ln(1/t_{break})$ as a function of $1/k_B T$ (green curve), with error bars representing the standard deviation. The red dashed line has slope -1. Except for the low temperature points where 10^4 time steps is not sufficient to capture the bond breaking time, data from the simulation is consistent with (2.28).

with valence 1, 2 and 3 respectively. The adjacency matrix is

$$\begin{bmatrix} 0 & 1 & 0.02 \\ 1 & 0 & 1 \\ 0.02 & 1 & 0 \end{bmatrix}, \tag{2.29}$$

i.e. strong bond with energy $E = 1$ can be formed between A, B and B, C. The crosstalking interaction has energy $\varepsilon = 0.02$, and the repelling strength is set to $\varepsilon_r = 1.5$. The system starts with 100 colloid particles, including 50 As, 30 Bs and 20 Cs. To approximate the grand canonical ensemble, we add new monomers into the system at random positions every $t_{add} = 300$ time steps to maintain the concentrations of different types of monomers in the system roughly constant, while uniformly remove assembled structures and monomers from the system with probability $p_{remove} = 0.15$ every $t_{remove} = 300$ time steps. We run the system at temperature $k_B T = 0.15$. Figure 2.9 again plots the system temperature as a function of the simulation time. It shows that the grand canonical ensemble particle system is also capable to cool down itself towards the input temperature. The system temperature eventually stays at a temperature acceptably higher than the input temperature due to the fact that the newly added monomers are set to random positions in the box, which might penetrate some solvents and result in strong repulsions. Figure 2.10 presents the composition of the system during the simulation process when particles are frequently added and removed from the box. Specifically, Figure 2.10A shows that the total mass of colloid in the system eventually reaches equilibrium at ~ 150 without keep increasing, and Figure 2.10B shows that the concentrations of different types of monomers are kept roughly constant. These verify that our simulated grand canonical ensemble is physically correct.

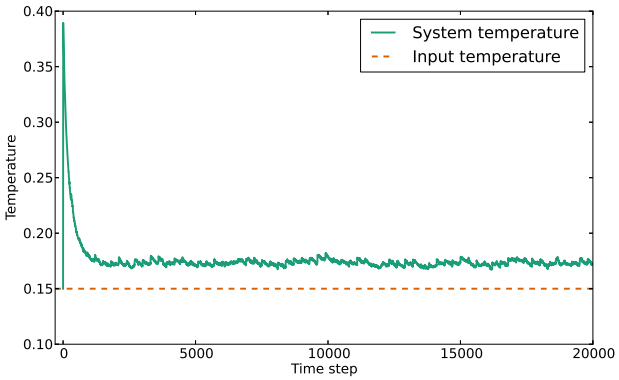


Figure 2.9: The system temperature (green curve) of the grand canonical ensemble particle system as a function of the simulation time. The simulated grand canonical ensemble particle system is also capable to cool down itself towards the input temperature (red dashed line).

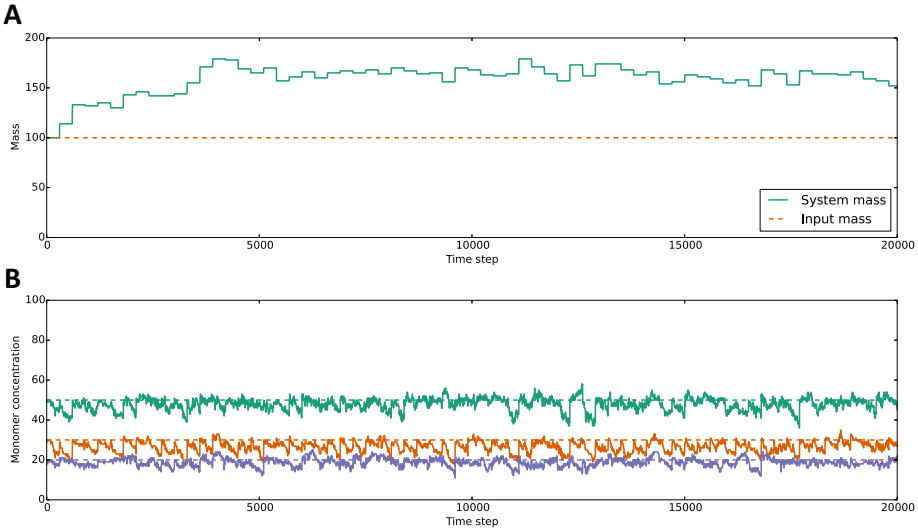


Figure 2.10: The composition of the simulated grand canonical ensemble particle system. **A** shows the total mass of colloid in the system (green curve), comparing to the original colloid mass 100 (red dashed line). The mass of colloid in the system eventually equilibrates at $\sim$ 150. In **B**, the dashed lines represent the input, or desired concentrations of A (green), B (red) and C (purple) monomers, and the solid curves are their actual concentrations in the simulation. Since we frequently add in new monomers into the system, their concentrations are maintained as desired.

2.4 Optimization Method I: Control the Interactions

To assemble a desired structure, one method is to choose the short-ranged interactions between neighbor components, so that the desired structure becomes the only ground state. Analytical elaboration[50] points out that to maximize the yield of a desired structure, it is advantageous to use more component types, or larger *alphabet*, than strictly required by local rules. In this chapter, we further demonstrate this with GPU-simulated particle system ensemble. Particularly, we simulate the assembly of colloid clusters with 6 particles.

2.4.1 Identical particle simulation

A rigid cluster of n_c identical colloid particles has degeneracy of the ground state when $6 \leq n_c \leq 10$[69,10,9], so that the equilibrium yield of different ground state clusters is determined by entropy. For $n_c = 6$, there are two ground states with 12 contacts: asymmetric polytetrahedron and symmetric octahedron, as Figure 2.11 shows. The rotational entropy strongly suppresses the yield of the symmetric octahedron. Theoretical calculation predicts that the yield of octahedron and polytetrahedron is 4% and 96% respectively[10,9]. This is confirmed by experiment[69].

We set the binding energy between particles to $\varepsilon_a = 1$, and run the GPU simulation from temperature 0.002 to 0.202 with step 0.002. For each temperature, an ensemble of 1000 independent particle systems are simulated simultaneously, and the states of the systems at 2×10^4 time steps are stored. Figure 2.12A shows the absolute yields of various structures as a function of the temperature. Figure 2.12B shows the relative yields of octahedron and polytetrahedron. It proves that our simulation result is consistent with the $96\% - 4\%$ theoretical prediction.

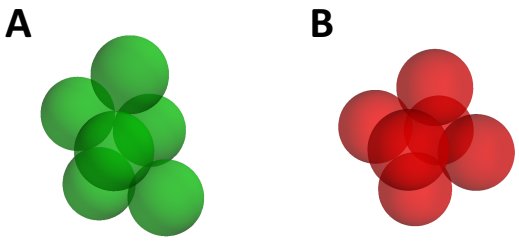


Figure 2.11: The two ground states of 6-particle rigid cluster: asymmetric polytetrahedron (**A**) and symmetric octahedron (**B**). Although they have the same potential energy as both of them have 12 particle-particle contacts, the rotational entropy strongly suppresses the yield of the octahedron. Theoretical calculation predicts that the yield of octahedron and polytetrahedron is 4% and 96% respectively.

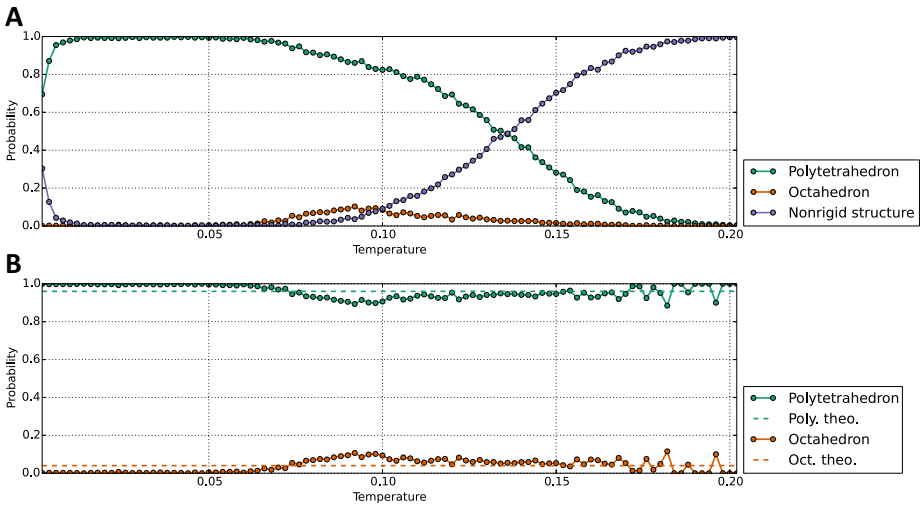


Figure 2.12: The yields of various structures as a function of the temperature for the 6-particle cluster assembly. In the simulation, we set $\varepsilon_a = 1$, and simultaneously simulate an ensemble of 1000 independent particle systems with 6 colloid particles. The states of the systems at 2×10^4 time steps are stored. **A** shows the absolute yields, or the probability that the final states of the systems is a specific structure. **B** shows the relative yields of octahedron and polytetrahedron. At temperature roughly between 0.11 and 0.16, the system reaches equilibrium, where the simulated yields are consistent with the theoretical predictions (dashed lines).

2.4.2 Colored particle simulation

When using colored colloid particles, interactions between particles can be carefully designed so that the ground state degeneracy is eliminated and only the desired conformation is allowed. For a specific structure, there exists a minimum number of colors, or letters, strictly required to break the degeneracy. Figure 2.13 shows all alphabets for the 6-particle clusters. The *interaction matrix* represents the attracting (1) and repelling (0) interactions between colors, while the *adjacency matrix* extends the interaction matrix to characterize the attraction (1) and repulsion (0) between all particle pairs in the structure. There is one alphabet favoring the octahedron, and two alphabets, one with 3 letters and the other with 5 letters, favoring the polytetrahedron.

Figure 2.14 shows the yield of various structures as a function of the temperature when using the octahedron alphabet. The attracting and repelling strength are $\varepsilon_a = 1$ and $\varepsilon_r = 1.5$ respectively. Since the octahedron is the only ground state, the yield of the polytetrahedron is suppressed to zero. Figure 2.15 shows the yield-temperature plots when using the 3-letter and 5-letter polytetrahedron alphabets. As expected, the yield of octahedron is suppressed to zero. Particularly, the polytetrahedron yield is higher when using 5-letter alphabet than 3-letter alphabet, as Figure 2.15C shows. This is consistent with the theoretical prediction[50] that using more component types increases the yield.

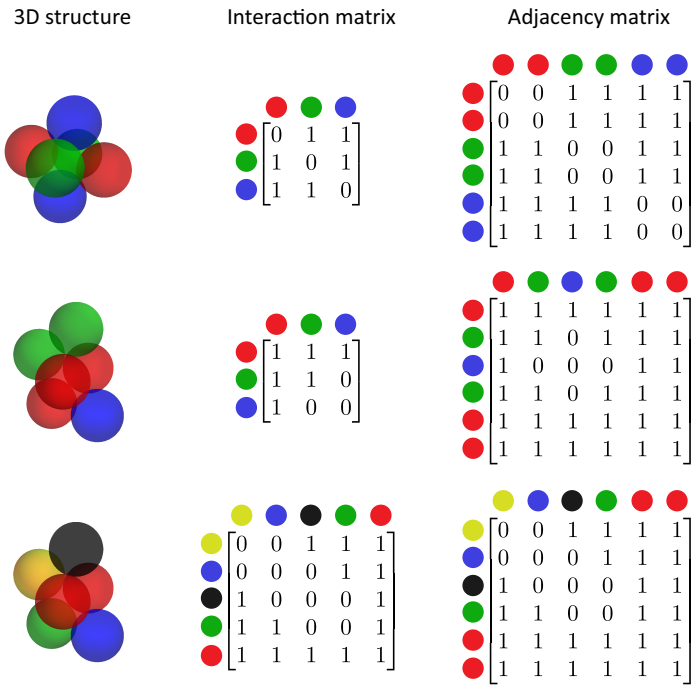


Figure 2.13: All alphabets for 6-particle clusters. The 1st column shows the colored 3D structure of the clusters. The 2nd column are the interaction matrices. They represent the attracting (1) and repelling (0) interactions between colors. The 3rd column shows the corresponding adjacency matrices. They are extended from the interaction matrix to characterize the interaction between all particle pairs in the structure. There is one alphabet favoring the octahedron (1st row), and two alphabets, one with 3 letters (2nd row) and the other with 5 letters (3rd row), favoring the polytetrahedron.

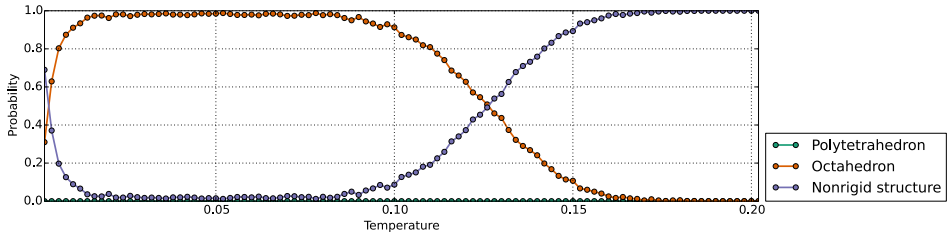


Figure 2.14: The absolute yield of various structures as a function of the temperature when using the octahedron alphabet. After applying the alphabet, the octahedron becomes the only ground state, therefore the yield of the polytetrahedron is suppressed to zero.

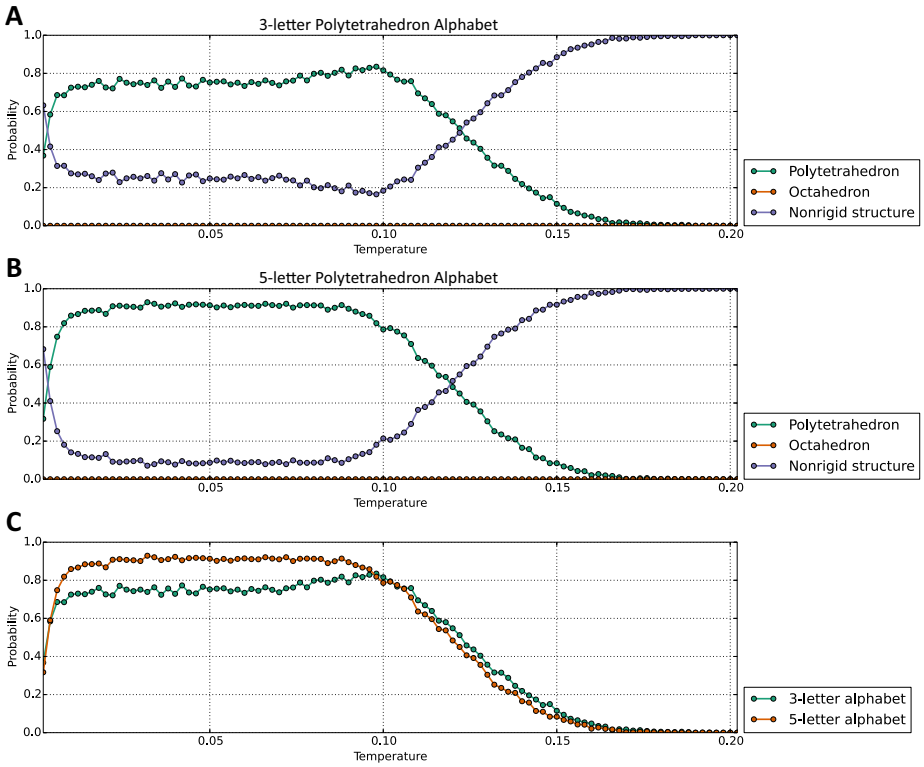


Figure 2.15: The absolute yield of various structures as a function of the temperature when using 3-letter (**A**) and 5-letter (**B**) polytetrahedron alphabets. The yield of octahedron is suppressed to zero since polytetrahedron is the only ground state cluster. **C** compares the yields of two alphabets. Consistent with the theoretical prediction, the 5-letter alphabet is more advantageous than the 3-letter alphabet.

2.5 Optimization Method II: Control the Concentrations

In the previous example, the self-assembly is performed in a system with the exact number of colloid particles in the desired structure. This is usually not the case as self-assembly often happens in a pool of large numbers of different components, and many structures are assembled in parallel in the same system. Therefore, an exponential number of partial and incorrect structures can form and compete with the desired structure. Analyses[70] conclude that the equilibrium yield of the desired structure can be improved by tuning the concentrations of different components, and the optimal concentration profile is highly non-uniform. Specifically, the authors analytically proves this using a 1D model where components have directional binding sites, and crosstalking interactions exist between all components. In this chapter, under a valence and crosstalk model, we demonstrate that the method of optimizing yield by tuning concentrations persists in 3D.

2.5.1 5-mer linear chain

We first demonstrate the concentration optimization by assembling a 5-mer chain. Figure 2.16A and B respectively shows the desired structure and the corresponding adjacency matrix used in the simulation. There are 5 types of colored particle in the structure. The valence of A and E is 1, while B, C and D have valence 2. Each type can only form strong bond(s) with its neighboring type(s), and crosstalking weak bonds can be formed between all types of particles, except particles of the same type.

The stoichiometry of the desired structure gives us the wrong intuition that the concentrations of different types of particles should be set equal in order to maximize its yield. In fact, the optimal concentration should take the particle consumptions in

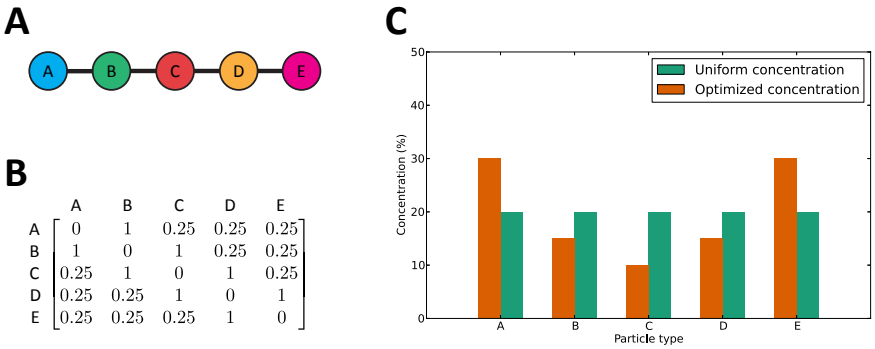


Figure 2.16: A shows the 5-mer chain and **B** shows the corresponding adjacency matrix used in the simulation. There are 5 types of particles. The valence of A and E is 1, and the rest particle types have valence 2. Strong bond(s) with the same energy 1 can be formed between neighboring types, and weak bonds with energy 0.25 due to crosstalking interactions can be formed between all different types of particles. **C** shows the uniform and the optimized U-shape concentration profile used in the simulation. The U-shape profile is derived from the reasoning that when a particle type is used more often in competing structures, its concentration should be suppressed to optimize the yield.

all competing structures, including partial and incorrect structures, into account. For example, if B is used more often in all competing structures than A, then A should have higher concentration than B in the optimal concentration profile. Consider an ideal case where there is no crosstalking interactions, by enumerating all structures that can be formed using the 5 particle types[†], A and E are used 4 times, B and C are used 7 times, and C is used 8 times in these structures. Therefore, the optimal concentration profile should be U-shape, where concentrations gradually decrease from the endpoints of the chain to the middle. Rigorous analysis for the U-shape optimal concentration profile can be found in the manuscript[70], where both partial structures and incorrect structures due to crosstalk are considered. Figure 2.16C shows the uniform and optimized concentration profiles we use in simulations.

[†]Without crosstalk, all possible structures are A-B, A-B-C, A-B-C-D, A-B-C-D-E, B-C, B-C-D, B-C-D-E, C-D, C-D-E and D-E.

Simulation on canonical ensemble

We first simulate the assembly using two canonical ensemble particle systems with 100 colloid particles. The composition of the systems are set according to the uniform and U-shape concentration respectively. The systems are simulated at temperature 0.13. We choose a relatively high temperature so that the system can sufficiently explore the energy landscape in the given simulation time. Snapshots of the system are taken every 10^3 time steps. Figure 2.17 shows the average yields of all structures at snapshots from 0.5×10^6 to 2×10^6 time steps, with color green representing partial structures, including the desired structure, and red representing incorrect structures due to crosstalking interaction. The tick labels of the vertical axis represent the composition of structures, but not necessarily the actual particle order. The U-shape concentration effectively suppresses the yields of BC and CD, and makes the yields of ABC, BCD, and CDE roughly equal. Because of the high temperature, the yields of large structures are small. However, if we zoom in to only examine the relative yields of 5-mer chain, the effect of the optimal concentration is obvious. As Figure 2.18 shows, the U-shape concentration significantly reduces the yields of competing 5-mer structures, and as a consequence, the yield of the desired structure stands out. As a more compact view, Figure 2.19 shows the number of structure types (left) and the average yields of the desired structure (right). The U-shape concentration effectively suppresses the emergence of competing types. The average yield of the desired structure is 4.15×10^{-4} under the U-shape concentration, comparing to 2.76×10^{-4} under the uniform concentration.

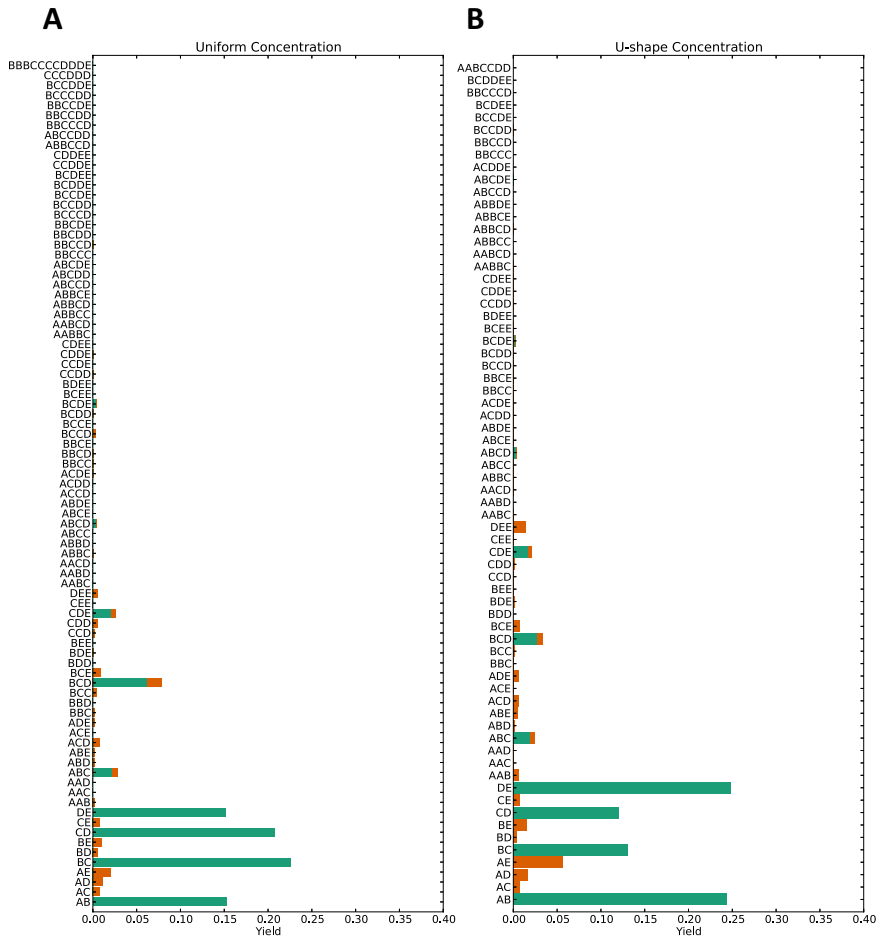


Figure 2.17: At temperature 0.13, the average yields of all structures in two canonical ensemble particle systems with 100 colloid particles, where concentrations of different types of particles are uniform (**A**) and U-shape (**B**) respectively. The average is taken over snapshots from 0.5×10^6 to 2×10^6 time steps, with 10^3 as step. The tick labels of the vertical axis represent the composition of structures, but not necessarily the actual particle order. Green bars represent partial structures, including the desired structure, whereas red bars represent incorrect structures as a result of the crosstalking interaction. The U-shape concentration effectively suppresses the yields of dimer BC and CD, and makes the yields trimer ABC, BCD, and CDE roughly equal. Because of the high temperature, the yields of large structures are small. The average yield of the desired structure is 0.000415 when using the U-shape concentration, comparing to 0.000276 for the uniform concentration

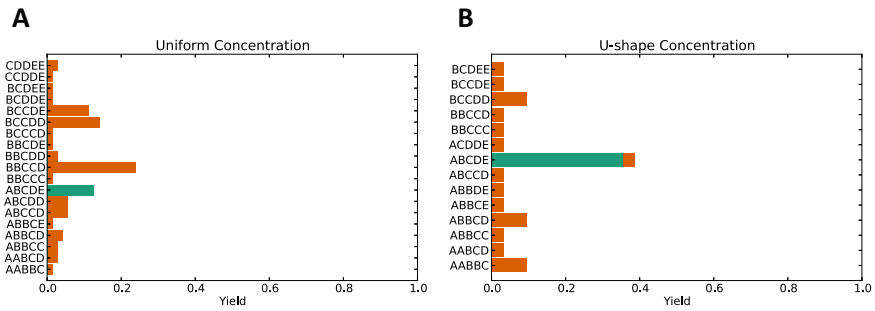


Figure 2.18: Relative yields of 5-mer structures in canonical ensemble particle systems, where concentrations of different types of particles are uniform (**A**) and U-shape (**B**) respectively. Green bars represent the desired structure, whereas red bars represent incorrect structures due to crosstalking interaction. The U-shape concentration effectively suppresses the yields of undesired structures, and makes the yield of the desired structure (green ABCDE) stand out.

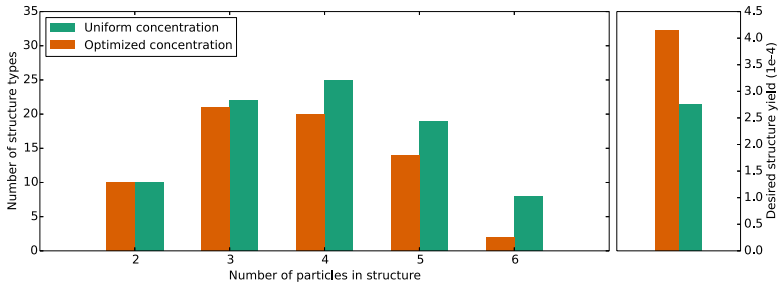


Figure 2.19: The number of structure types (left) and the average yields of the desired structure (right) when simulating the 5-mer chain assembly with canonical ensemble particle systems. The U-shape concentration effectively suppresses the emergence of competing types. The average yield of the desired structure is 4.15×10^{-4} under the U-shape concentration, comparing to 2.76×10^{-4} under the uniform concentration.

Simulation on grand canonical ensemble

We then simulate the assembly using grand canonical ensemble particle systems with 100 colloid particles. We add new monomers into the system every $t_{add} = 200$ time steps to maintain the monomer concentrations as desired, and remove structures from the system with probability $p_{remove} = 0.15$ every $t_{remove} = 800$ time steps. The system again runs at temperature 0.13. Figure 2.20 shows the yields of structures removed from the systems between 0.5×10^6 and 10^6 time steps, and Figure 2.21 shows the relative yields of 5-mer structures. The effect of the U-shape concentration is similar to the canonical ensemble case: the yields of partial and incorrect structures are significantly suppressed, and the yield of the desired structure stands out. Figure 2.22 shows the number of structure types (left) and the average yields of the desired structure (right). Similarly, the number of competing structure types appear less when using the optimized concentration. The average yield of the desired structure is 8.17×10^{-4} when using the U-shape concentration, comparing to 1.85×10^{-4} for the uniform concentration.

2.5.2 Protein complex

We also simulate the assembly of a real protein complex. Figure 2.23A and B respectively shows its 3D and 2D structure. It is a transferase composed of 6 components[82]. They are approximated by round particles in our simulation. We set the valence of A, B and C to 3, and the valence of D, E and F to 1. Figure 2.23C shows the adjacency matrix. The binding energies of strong bonds used in the simulation are proportional to the actual binding energy in Figure 2.23B, and the energy of the crosstalking interactions is set to 0.1. Figure 2.23D shows the uniform and the optimized concentrations we use

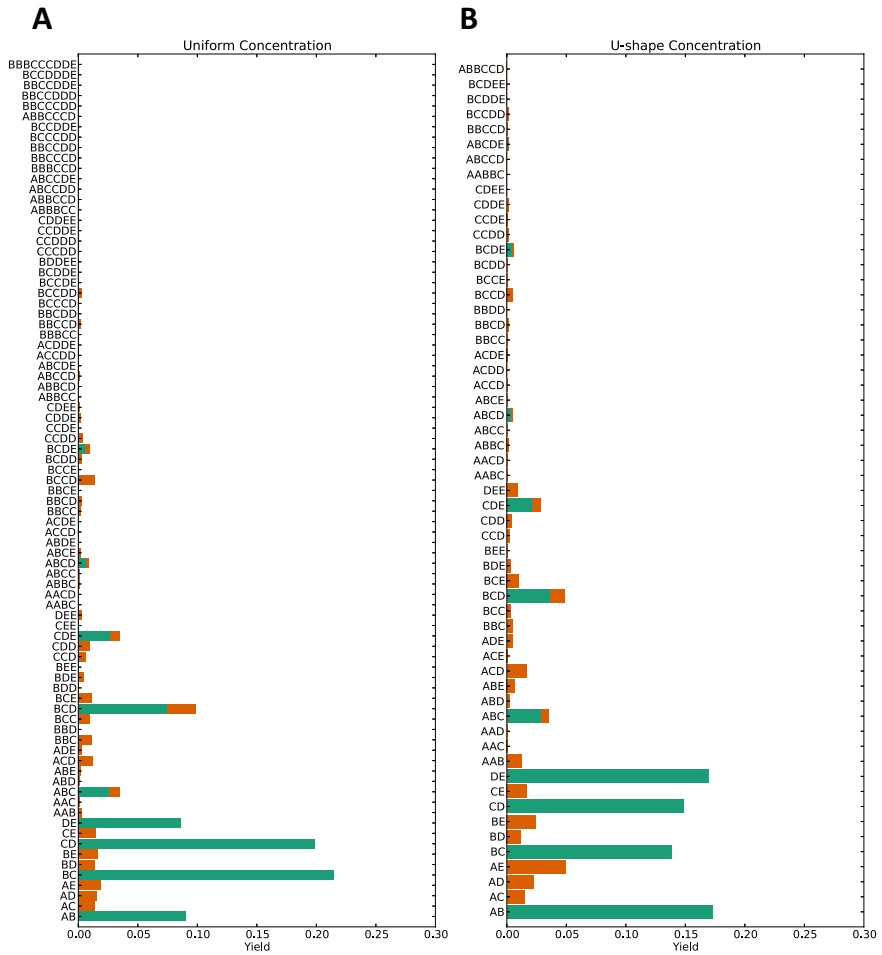


Figure 2.20: At temperature 0.13, the average yields of all structures in two grand canonical ensemble particle systems with 100 initial colloid particles, whose concentrations of different types of particles are maintained at uniform (**A**) and U-shape (**B**) respectively by frequently adding new monomers into the systems. Green bars represent partial structures, including the desired structure, whereas red bars represent incorrect structures as a result of the crosstalking interaction. The U-shape concentration effectively suppresses the yields of dimer BC and CD, and makes the yields trimer ABC, BCD, and CDE roughly equal.

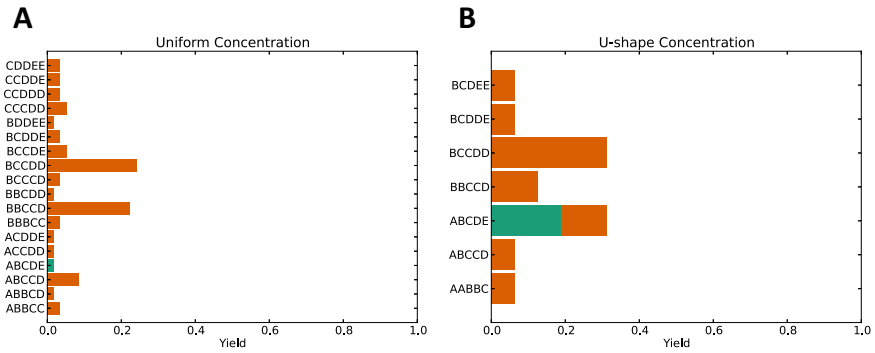


Figure 2.21: Relative yields of 5-mer structures in grand canonical ensemble particle systems where concentrations of different types of particles are maintained at uniform (**A**) and U-shape (**B**). Green bars represent the desired structure, whereas red bars represent incorrect structures due to crosstalking interaction. The U-shape concentration significantly suppresses the yields of incorrect structures, and makes the yield of the desired structure (green ABCDE) stand out.

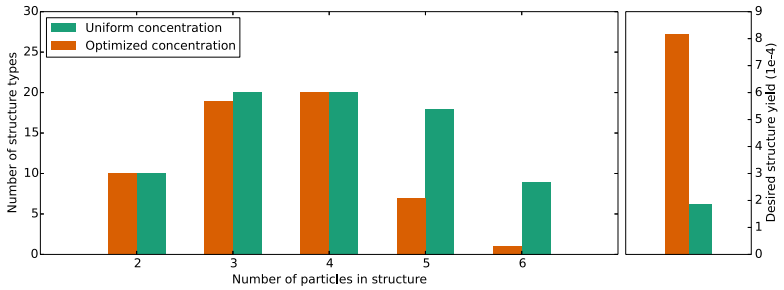


Figure 2.22: The number of structure types (left) and the average yields of the desired structure (right) when simulating the 5-mer chain assembly with grand canonical ensemble particle systems. The U-shape concentration effectively suppresses the emergence of competing types. The average yield of the desired structure is 8.17×10^{-4} under the U-shape concentration, comparing to 1.85×10^{-4} under the uniform concentration.

in the simulation. The optimized concentration suppresses the concentrations of the inner particles (A, B and C) and enlarges the concentrations of the branching particles (D, E and F). In the canonical ensemble simulation, snapshots of the systems are taken every 1000 time steps between 0.5×10^6 to 2×10^6 time steps. Figure 2.24 shows the average relative yields of all 6-mer structures in two canonical ensemble systems with 120 colloid particles, whose concentrations of different particle types are uniform and optimized respectively. When using the optimized concentration, in total there are only 10 types of 6-mer structures appearing in the snapshots, comparing to 32 types of 6-mer structures for the uniform concentration. Particularly, structures containing 2 As, 2 Bs and 2 Cs are dominant under the uniform concentration, whereas when using the optimized concentration, they are significantly suppressed, and the yield of the desired structure (green ABCDEF) is much larger. Figure 2.25 shows the number of structure types (left) and the average yields of the desired structure (right). The optimized concentration effectively suppresses the emergence of competing types. The average yield of the desired structure is 1.61×10^{-4} when using the optimized concentration, comparing to 0.616×10^{-4} under the uniform concentration.

2.5.3 Finite time behavior

All previous analyses focus on the equilibrial behavior of a self-assembly system. On the other hand, since equilibrium is not guaranteed in some cases, it is also important to study the impact of the concentration, optimized for equilibrial yield, on finite time yield of the system. For this purpose, we simulate an ensemble of particle systems that self-assemble the the 6-particle protein complex (Figure 2.23).

Figure 2.26 shows the average yields of partial and desired structures, taken over 200 independent canonical ensemble particle systems, as a function of time. Each sys-

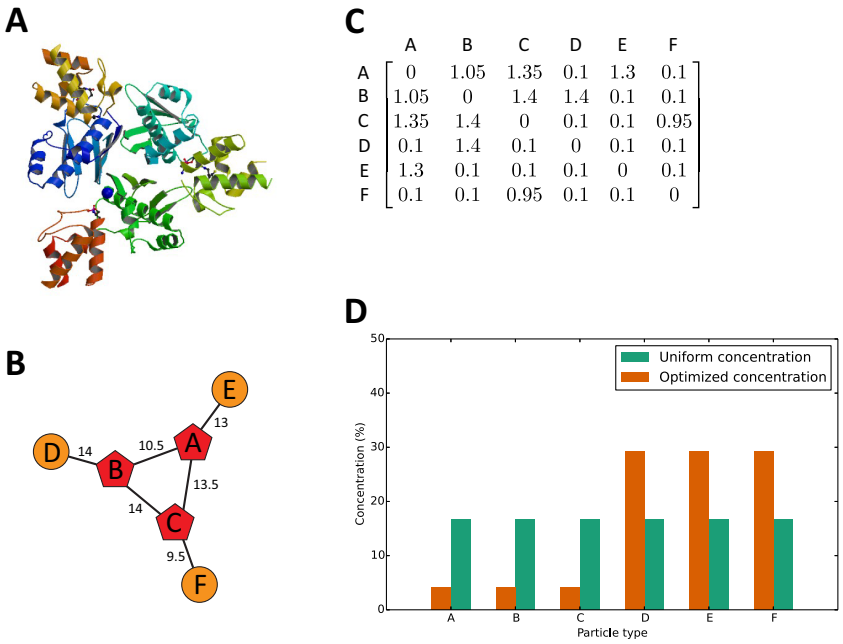


Figure 2.23: Snapshots of the transferase protein with 6 components. **A** shows the 3D folding structure taken from the RCSB Protein Data Bank [15]. **B** shows its 2D structure, reflecting schema on the 3D Complex website [64], along with the strengths of the bonds. **C** is the adjacency matrix. The binding energies of strong bonds used in the simulation are proportional to the actual binding energy in **B**. **D** shows the uniform and the optimized concentrations used in the simulation. The optimized concentration suppresses the concentrations of the inner particles A, B and C, while enlarging the concentrations of the branching particles D, E and F.

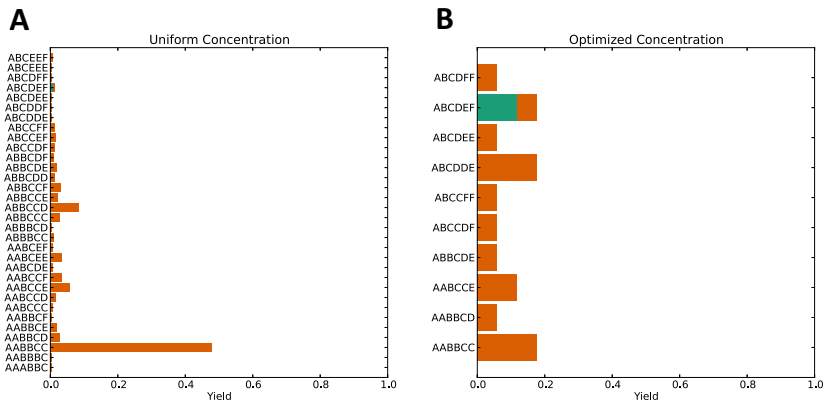


Figure 2.24: The average relative yields of all 6-mer structures in two canonical ensemble systems with 120 colloid particles, whose concentrations of different particle types are uniform (**A**) and optimized (**B**) respectively. Green bars represent the desired structure, while red bars are undesired structure due to crosstalk. When using the optimized concentration, in total there are only 10 types of 6-mer structures appearing in the snapshots, comparing to 32 types of 6-mer structures for the uniform concentration. Particularly, structures containing 2 As, 2 Bs and 2 Cs are dominant under the uniform concentration, whereas under the optimized concentration, they are strongly suppressed, and the yield of the desired structure (green ABCDEF) is much larger.

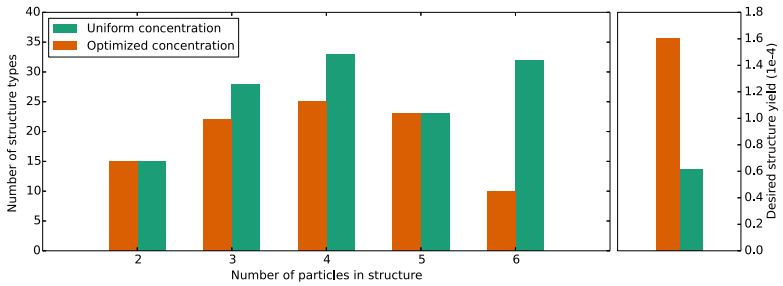


Figure 2.25: The number of structure types (left) and the average yields of the desired structure (right) when simulating the protein complex assembly with canonical ensemble particle systems. The optimized concentration effectively suppresses the emergence of competing types. The average yield of the desired structure is 1.61×10^{-4} when using the optimized concentration, comparing to 0.616×10^{-4} under the uniform concentration.

tem has 1200 particles. Snapshots of the systems are taken every 500 time steps, and the yield is calculated as the number of a particular structure divided by the number of all structures, excluding monomers, at the snapshot. Dots represent the average yield at snapshots, while solid curves are moving averages over a window of 10 snapshots, or 5000 time steps. As we can see, yields of different structures gradually increase over time, and fix at certain levels after equilibrium. Systems with uniform concentration reach equilibrium at $\sim$ 40000 time steps, faster than systems with optimized concentration, which take $\sim$ 60000 time steps to reach equilibrium. Since the optimized concentration suppresses the concentrations of the inner particles A, B and C, the yield of the triangle ABC is $\sim$ 2.5 times lower after concentration optimization, as Figure 2.26A shows. Figure 2.26B shows that although the yield of a 4-mer structure, composed of the inner triangle and a branching particle D, is lower in systems with optimized concentration than those with uniform concentration, their gap is smaller compare to Figure 2.26A. Figure 2.26C shows that before $\sim$ 40000 time steps, the yield of a 5-mer structure, composed of the inner triangle and two branching particles, is lower in systems with optimized concentration, but slightly surpasses its uniform counterpart after that. Figure 2.26D shows the yields of the desired structure. Before $\sim$ 40000 time steps, both concentration profiles produce similar yield, but systems with optimized concentration eventually achieve significantly higher yield when they reach equilibrium. To summarize, the optimized concentration enhances the yield of the desired structure by suppressing the yields of smaller partial structures, while enlarging the yields of larger partial structures.

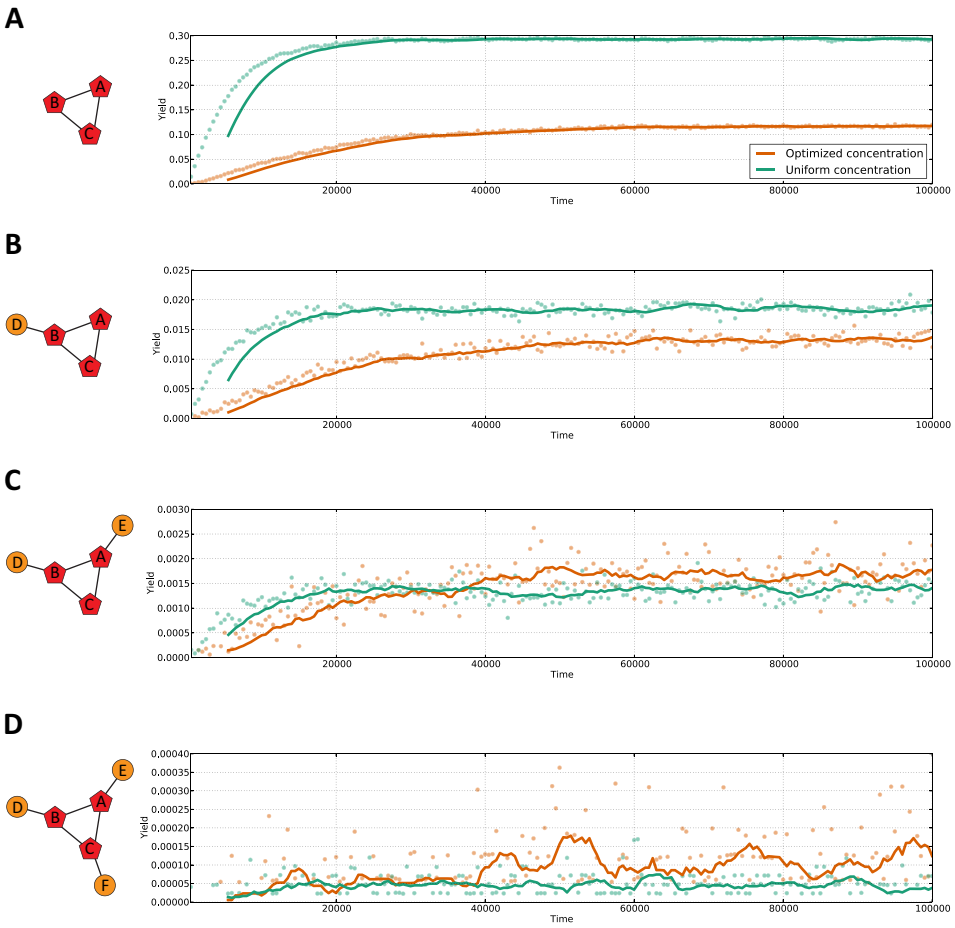


Figure 2.26: The average yields of partial and desired structures, taken over 200 independent canonical ensemble particle systems, as a function of time. Each system has 1200 particles. Snapshots of the systems are taken every 500 time steps, and the yield is calculated as the number of a particular structure divided by the number of all structures, excluding monomers, at the snapshot. Dots represent the average yield at snapshots, while solid curves are moving averages over a window of 10 snapshots, or 5000 time steps. Yields of different structures gradually increase over time, and fix at certain levels after equilibrium. Systems with uniform concentration reach equilibrium at $\sim$ 40000 time steps, faster than systems with optimized concentration, which take $\sim$ 60000 time steps to reach equilibrium. The optimized concentration enhances the yield of the desired structure by suppressing the yields of smaller partial structures, while enlarging the yields of larger partial structures.

2.6 Summary

This chapter presents simulation results of two self-assembly optimization methods using GPU computing. Both results are consistent with theoretical predictions. When controlling the short-ranged interactions between particles, it is advantageous to use more particle types than strictly required by local rules. When controlling the concentrations of monomers, counterintuitive nonuniform concentrations enhance the yield of the desired structure. Particularly, nonequilibrium simulation shows that particle system with optimized concentration reaches equilibrium slower than its uniform counterpart, and the optimized concentration enhances the yield of the desired structure by suppressing the yields of smaller partial structures, while enlarging the yields of larger structures.

Since long integration time and numerous data are necessary to ensure equilibrium and accurate statistics, GPU computing is a natural and powerful choice to study self-assembly optimization problems given the parallel nature of the Verlet integration steps. As the GPU computing momentum continues to grow exponentially fast, it will certainly play an increasingly important, if not essential, role in self-assembly and many other fields in the near future.

I have not failed. I've just found 10000 ways that won't work.

Thomas Edison

3

Signal Detection below the Random Matrix Theory Threshold

A critical question when analyzing multivariate datasets is to understand the correlations between variables, and their clustering so forth. Random Matrix Theory (RMT)[12,32,52] is an efficient set of methodologies to separate correlation signal from noise by analyzing eigenvalues and eigenvectors of the sample correlation matrix. In

this chapter, using RMT as a benchmark, I explore the detection power of a more general criterion, the *likelihood function*. More specifically, an algorithm based on *simulated annealing*[59,97] is proposed to efficiently traverse the likelihood landscape and search for the optimal true correlation matrix with blocks. Further analysis shows that the algorithm is capable to detect signals below the RMT threshold after incorporating some existing information into the search process. The efficacy of the algorithm is tested on both synthetic and real financial datasets. A homogeneous model is also proposed to explain the phenomenon.

The structure of the chapter is as follows. Section 3.1 gives an overview of the Random Matrix Theory and its application in signal detection. Section 3.2 elaborates the proposed signal detection algorithm. Section 3.3 presents simulation results on synthetic datasets with various signal patterns. Section 3.4 introduces the methodology of incorporating prior information into the algorithm to enable detections below the RMT threshold. Section 3.5 shows the results of applying the algorithm on real financial dataset. Section 3.6 presents a homogeneous model to quantify the detection power of the algorithm. Section 3.7 summarizes the chapter.

3.1 Random Matrix Theory

RMT is a set of methodologies that have been applied to various fields for signal detection, such as finance[61,86], bioinformatics[28] and communications[95]. In this section, we review its two important conclusions.

3.1.1 The Marčenko-Pastur law

One of the most important theories in RMT is the Marčenko-Pastur (M-P) law[67]. It characterizes the eigenvalue distribution, or the *spectrum*, of the sample correlation matrix of a multivariate dataset. Suppose there are p variables, each measured by n samples. We can gather the data into a normalized p-by-n data matrix X, whose rows represent variables and columns represent samples. Each row of X is normalized to mean 0 and standard deviation 1. The sample correlation matrix is then calculated as

$$S = \frac{XX^T}{n}. \tag{3.1}$$

Denote the sorted eigenvalues of S as $\{\lambda_i \,|\, i = 1, 2, \ldots, p,\ \lambda_1 > \lambda_2 > \cdots > \lambda_p\}$. The M-P law states that when all variables are independent, identically distributed (i.i.d.) with variance 1 and fourth moment of order $\mathcal{O}(1)$, in the limit of

$$p, n \to +\infty, \quad \frac{p}{n} \to \beta < +\infty, \tag{3.2}$$

the limiting distribution of λ is

$$f\ \ (x) = \left(1 - \frac{1}{\beta}\right)^+ \delta(x) + \frac{\sqrt{(x - \lambda_{min})^+ (\lambda_{max} - x)^+}}{2\pi\beta x}, \tag{3.3}$$

where $\lambda_{min} = (1 - \sqrt{\beta})^2$, $\lambda_{max} = (1 + \sqrt{\beta})^2$, $\delta(x)$ is the Dirac delta function, and the notation $x^+ = \max(x, 0)$.

3.1.2 Gaussian eigenvectors

RMT also characterizes eigenvectors of the sample correlation matrix. Suppose each eigenvector is normalized to norm 1. Since the dataset does not contain correlation information, in the limit (3.2), the limiting distribution of eigenvector components follow the maximum entropy distribution, i.e. the normal distribution, with mean zero and standard deviation $1/\sqrt{p}$:

$$f(x) = \sqrt{\frac{p}{2\pi}} \exp\left(-\frac{px^2}{2}\right). \tag{3.4}$$

3.1.3 RMT based signal detection

Although most RMT theorems hold asymptotically in the limit (3.2), they are almost always applicable to finite p and n. As a demonstration, we generate a random Gaussian dataset with $p = 300$ variables and $n = 400$ samples and compute its sample correlation matrix. Figure 3.1 shows the spectrum and the distribution of components of the 1st eigenvector. The spectrum is consistent with the M-P law, and the eigenvector components are Gaussian.

When there are correlated variables in the dataset, the M-P law can be used as a null hypothesis to separate nontrivial eigenvalues from trivial eigenvalues, and the actual signals can be identified from the corresponding eigenvectors. Figure 3.2 illustrates the RMT based signal detection procedure. It shows a case where there are two blocks of correlated variables in a Gaussian dataset with $p = 300$ variables and $n = 400$ sam-

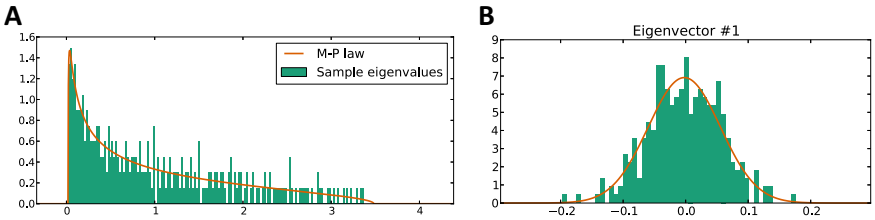


Figure 3.1: Spectrum (**A**) and the 1st eigenvector components (**B**) of the sample correlation matrix of a random Gaussian dataset with $p = 300$ variables and $n = 400$ samples. In **A**, the spectrum (green bars) is consistent with the M-P law (red curve). Specifically, all eigenvalues are inside the edges (λ_{min} and λ_{max} in (3.3)) of the M-P law. **B** shows that components of the 1st eigenvector (green bars) are consistent with normal distribution (red curve).

ples. Figure 3.2A and B visualizes the true and sample correlation matrix respectively. Figure 3.2C plots the spectrum (green bars) and the M-P law (red curve). There are two eigenvalues above the upper edge of the M-P law, indicating two blocks of correlated variables. The actual clusters can be identified by examining eigenvectors associated to nontrivial eigenvalues. As Figure 3.2D shows, the 1st and 2nd eigenvectors significantly deviate from Gaussian. Specifically, each of them has an obvious clusters of components, corresponding to the two blocks of correlated variables.

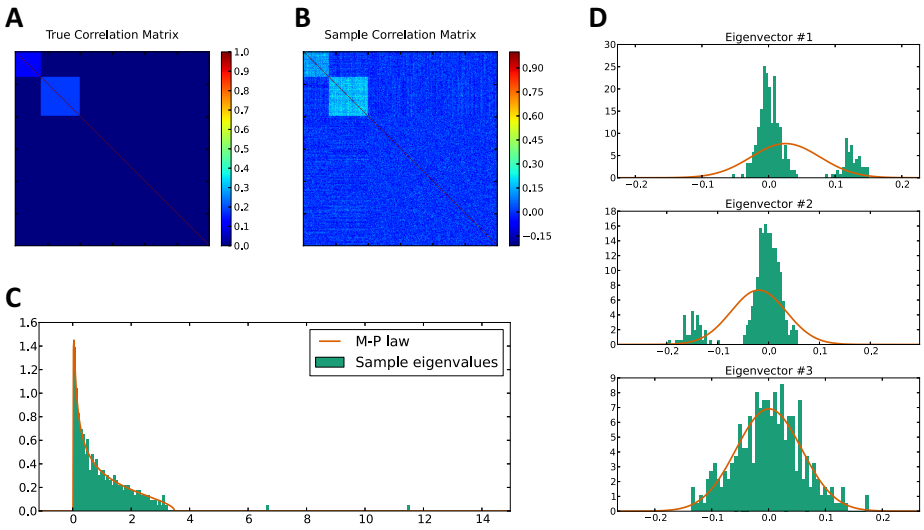


Figure 3.2: The RMT based signal detection procedure. We generate a Gaussian dataset with 300 variables, 400 samples. There are two blocks of correlated variables. **A** and **B** visualizes the true and sample correlation matrices respectively. **C** plots the spectrum of the sample correlation matrix. There are two signal eigenvalues above the upper edge of the M-P law, indicating two blocks of correlated variables. **D** shows the top three eigenvectors. Each of the 1st and 2nd eigenvector has an obvious cluster, respectively corresponding to the two blocks of correlated variables. Components of the 3rd eigenvector distribute as Gaussian, inferring that there is no information left from this eigenvector.

3.2 Likelihood Based Signal Detection

Since we are interested in the optimal correlation matrix C giving rise to the p-by-n data matrix X, a natural choice is to evaluate the *likelihood function* $\mathcal{L}$ of C, and search for the correlation matrix C^* with the *maximum likelihood*, given the data matrix and other prior information:

$$C^* = \text{argmax}_C \ \mathcal{L}(C \mid X, \text{prior information}). \tag{3.5}$$

The prior information could be the underlying distribution of the variables, as well as structural knowledge of C. Though not necessary, analyses in this chapter are restricted to multivariate normal distribution.

3.2.1 Likelihood function of multivariate normal distribution

The likelihood function of multivariate normal distribution is

$$\mathcal{L} = (2\pi)^{-\frac{pn}{2}} \det(C)^{-\frac{n}{2}} \exp\left(-\frac{1}{2}\sum_i^n x_i^T C x_i\right), \tag{3.6}$$

where x_i is the i-th realization of the variables, or the i-th column of X. Logarithm of the likelihood, or the *log-likelihood*, is often considered for algebraic simplicity. The log-likelihood of multivariate normal distribution is

$$\ln(\mathcal{L}) = -\frac{pn}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(C)) - \frac{1}{2}\sum_{i=1}^n x_i^T C^{-1} x_i. \tag{3.7}$$

Without other information, it is widely known that the sample correlation matrix S

maximizes the likelihood:

$$C^* = \text{argmax}_C \, \mathcal{L}(C \,|\, X, \text{Gaussian variables}) = S. \tag{3.8}$$

However, pattern usually does not surface in the sample correlation matrix, especially when the signal is weak. If given some prior structural information of the correlation matrix, cleaner correlation pattern could be extracted.

3.2.2 Correlation matrix with blocks

Correlation matrix with blocks attracts practical interests, as it indicates clusters of correlated variables, or categorization of the population. In particular, we consider correlation matrices with independent blocks, and correlations within each block are uniform:

$$C = \begin{bmatrix} C_1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & C_k & 0 \\ 0 & \cdots & 0 & I \end{bmatrix}, \tag{3.9}$$

where C_i $(i = 1, \ldots, k)$ represents a submatrix with 1 on the diagonal and a uniform value for all off-diagonal components. The matrix in Figure 3.2A is an example of $k = 2$.

Incorporating the block correlation structure as prior information, the problem becomes searching for

$$C^* = \text{argmax}_C \, \mathcal{L}(C \,|\, X, \text{Gaussian variables, C has block structure}). \tag{3.10}$$

If the number of blocks is not given a priori, there could be at most p blocks in the correlation matrix, and each variable could belong to any of them. As a result, the size of the search space is $\mathcal{O}(p^p)$. It is unpractical to exhaustively scan over this gigantic space. Hence, a more intelligent search algorithm is necessary.

3.2.3 The search algorithm

Efficient computation of the log-likelihood

The nominal computation complexity of the log-likelihood (3.7) is $\mathcal{O}(p^3)$. However, we can take advantage of the low dimensional structure of C to efficiently compute the determinant and inverse in (3.7).

Lemma 1. *The determinant of a matrix with block structure equals the product of the determinants of the blocks; the inverse of a matrix with block structure is another matrix with block structure, whose blocks are the inverse of the corresponding blocks in the original matrix:*

$$\det \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_k \end{bmatrix} = \prod_{i=1}^{k} \det(C_i); \tag{3.11}$$

$$\begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_k \end{bmatrix}^{-1} = \begin{bmatrix} C_1^{-1} & 0 & \cdots & 0 \\ 0 & C_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_k^{-1} \end{bmatrix}. \tag{3.12}$$

Since each block has 1 on the diagonal and the off-diagonal components are a uniform constant, we can calculate its determinant and inverse in constant time according to the following lemma.

Lemma 2. *The determinant and inverse of an h-by-h matrix with 1 on the diagonal and α on the off-diagonal can be calculated in constant time:*

$$\det \begin{bmatrix} 1 & \alpha & \cdots & \alpha \\ \alpha & 1 & \cdots & \alpha \\ \vdots & \vdots & \ddots & \vdots \\ \alpha & \alpha & \cdots & 1 \end{bmatrix} = (1-\alpha)^{h-1}[(h-1)\alpha + 1]; \tag{3.13}$$

$$\begin{bmatrix} 1 & \alpha & \cdots & \alpha \\ \alpha & 1 & \cdots & \alpha \\ \vdots & \vdots & \ddots & \vdots \\ \alpha & \alpha & \cdots & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \beta & \gamma & \cdots & \gamma \\ \gamma & \beta & \cdots & \gamma \\ \vdots & \vdots & \ddots & \vdots \\ \gamma & \gamma & \cdots & \beta \end{bmatrix}, \tag{3.14}$$

where

$$\beta = \frac{(2-h)\alpha - 1}{[(h-1)\alpha + 1](\alpha - 1)}, \quad \gamma = \frac{\alpha}{[(h-1)\alpha + 1](\alpha - 1)}.$$

With Lemma 1 and 2, we can algebraically factorize the log-likelihood (3.7) into a block-wise summation:

$$\ln(\mathcal{L}) = -\frac{pn}{2}\ln(2\pi) - \frac{n}{2}\sum_{i=1}^{k}\{\ h_i + (1-\alpha_i)^{h_i-1}[(h_i-1)\alpha_i + 1]\ \}, \tag{3.15}$$

where k is the number of blocks in C, h_i and α_i are the width and off-diagonal value

of C_i respectively. In the simulation, α_i is approximated by the average sample correlations within reconstructed block. The computation complexity of (3.15) is $\mathcal{O}(k)$, much smaller than the nominal complexity $\mathcal{O}(p^3)$. In line with the convention of optimization problems, we further transform (3.15) by removing the constant term and the negative coefficient to define an energy

$$E = \sum_{i=1}^{k} E_i = \sum_{i=1}^{k} \{\ h_i + (1-\alpha_i)^{h_i-1}[(h_i-1)\alpha_i + 1]\ \}, \tag{3.16}$$

where E_i is the energy of the i-th block. Maximizing the log-likelihood is equivalent to minimizing E.

Simulated annealing

We use *simulated annealing*[59] to execute the search. In the search process, we randomly transform the correlation matrix, then accept or reject the transformation according to the *Metropolis acceptance rule*. Specifically, the probability of accepting a transformation from a correlation matrix with energy E_t to another with E_{t+1} is

$$Pr = \begin{cases} \exp(-\Delta E/T) & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases} \tag{3.17}$$

where $\Delta E = E_{t+1} - E_t$, and T is an artificial *temperature*, which is gradually decreased during the search process. This formula was superficially justified by analogy with the transitions of a physical system: it corresponds to the *Metropolis-Hastings algorithm*[45], in the case where the proposal distribution of Metropolis-Hastings is symmetric. For large Ts, the algorithm scans widely on the energy landscape and accepts more unfavor-

able moves ($\Delta E > 0$), whereas for small Ts, the algorithm searches locally and rejects most unfavorable moves.

Correlation matrix transformation

The last missing piece of the algorithm is the random transformation of the correlation matrix. We randomly generate a correlation matrix with block structure as the initial guess. Later on the algorithm executes the following three transformation methods one by one in the search:

1. *Shuffling.* Change the block each variable currently belongs to to another randomly chosen block.
2. *Merging.* Merge two randomly chosen blocks.
3. *Splitting.* Split a randomly chosen block into two blocks at a random dividing point.

Note that in all of the above transformation methods, the algorithm only change the compositions of two blocks. Since the energy of the whole correlation matrix is the summation of energies of all blocks (3.16), we only need to calculate the energy changes of these two blocks in order to calculate the energy change of the whole correlation matrix, leading to a further optimization of the computation efficiency. In the following text, we denote a set of these three transformations one by one, with the Metropolis acceptance rule applied to each transformation, as an *iteration* of matrix transformation.

3.3 Simulation Results

Our algorithm is programmed in C++ to leverage its high speed. In the simulations, we gradually decrease the temperature T from 0.025 to 0.00625. 4×10^5 iterations of matrix transformations are executed for each temperature. The goal is to obtain quality reconstructions of the compositions of blocks. To precisely evaluate the reconstruction, we introduce an *overlap score*. Specifically, let B and B' be an original block and a reconstructed block respectively. They are sets of variable indices. We define their overlap score as

$$\text{overlap score} = \frac{2|B \cap B'|}{|B| + |B'|}. \tag{3.18}$$

This is a score between 0 and 1, with 0 representing zero overlap and 1 perfect overlap. Since it is unclear which original block a reconstructed block corresponds to, for each reconstructed block, we calculate its overlap scores with all original blocks, and use the best score to represent its quality. We test our algorithm in three scenarios.

3.3.1 Uniform blocks with strong signal

We first construct a Gaussian data matrix with $p = 300$ variables and $n = 400$ samples, whose underlying 300-by-300 true correlation matrix has three independent, uniform and blocks with high correlations, as Figure 3.3A illustrates. Figure 3.3B shows the spectrum. The top three eigenvalues, corresponding to the three signal blocks, are significantly above the upper edge of the M-P law. Therefore, signals can be reconstructed using RMT. Figure 3.3C is the correlation matrix reconstructed by our method. Comparing to Figure 3.3A, both the sizes and strengths of the blocks are reconstructed almost perfectly. Figure 3.3D shows the best overlap scores of the reconstructed blocks.

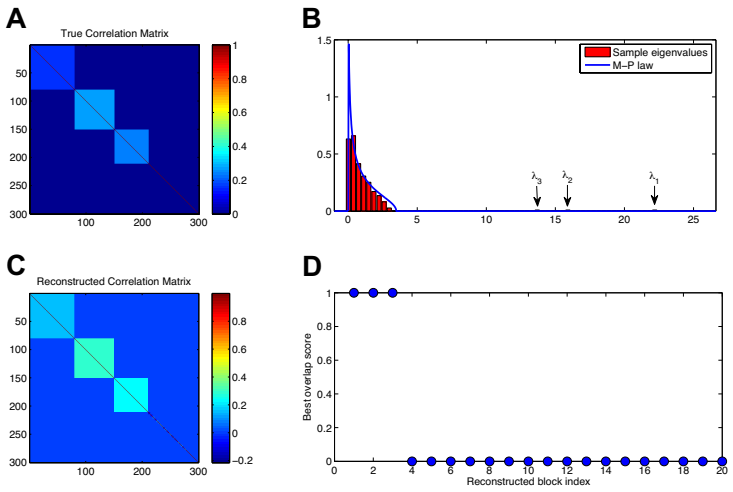


Figure 3.3: Simulation on a synthetic 300-by-400 Gaussian data matrix, whose underlying true correlation matrix has three uniform and strong blocks. **A** visualizes the true correlation matrix. **B** shows the eigenvalue distribution of the sample correlation matrix. Its top three eigenvalues are significantly above the upper edge of the M-P law. **C** shows the reconstructed correlation matrix by our method. Both sizes and strengths of the blocks are reconstructed almost perfectly. **D** plots the best overlap score of the reconstructed blocks. Three reconstructed blocks have perfect overlap scores ($= 1$), while others have near-zero scores, indicating that the algorithm effectively detects the signals.

Three reconstructed blocks have perfect overlap scores ($= 1$), while others have near-zero scores, indicating that the algorithm effectively separates signals from noise.

3.3.2 Nonuniform blocks with strong signal

We then test a scenario more similar to practical data analysis, where simplified models are fitted to datasets with a more complicated underlying distribution. We construct a Gaussian data matrix with $p = 300$ variables and $n = 400$ samples, whose true correlation matrix is visualized in Figure 3.4A. It contains 7 independent blocks, and correlations within each block are uniformly random in $[0, 0.3]$. Figure 3.4B shows the spectrum of the sample correlation matrix. There are 7 nontrivial eigenvalues above the upper edge of the M-P law, corresponding to the 7 blocks. Despite the underlying

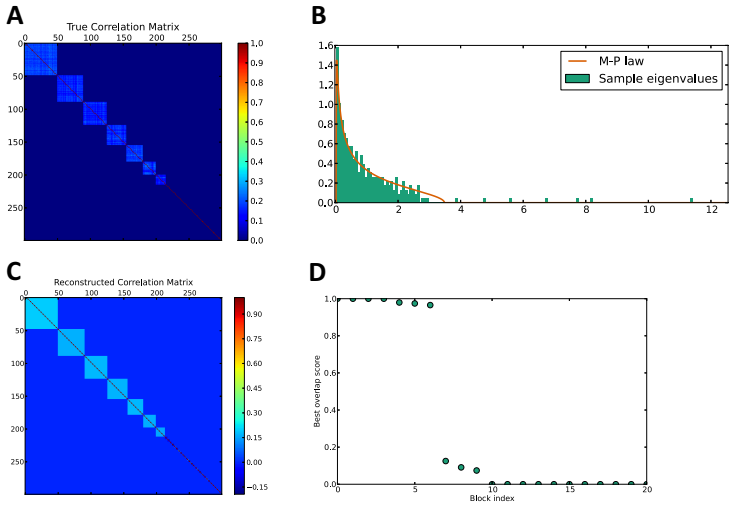


Figure 3.4: Simulation on a synthetic 300-by-400 Gaussian data matrix, whose underlying true correlation matrix has 7 nonuniform blocks. Correlations within each blocks are uniformly random in $[0, 0.3]$, as **A** shows. **B** shows the spectrum of the sample correlation matrix. There are 7 nontrivial eigenvalues above the upper edge of the M-P law. **C** shows the reconstructed correlation matrix, which is obtained by applying our algorithm assuming uniform blocks to the dataset. **D** shows the best overlap scores of the reconstructed blocks. Although the algorithm cannot unveil the structures of the original blocks, compositions of the blocks are almost perfectly reconstructed.

nonuniform block structure, we still use our method, which assumes uniform blocks, to perform the reconstruction. Figure 3.4C shows the reconstructed correlation matrix, where blocks are uniform, and Figure 3.4D shows the scores of the reconstructed blocks. They show that although the algorithm is uninformed of the structures of the original blocks, it almost perfectly identifies the compositions of the blocks.

3.3.3 Uniform blocks with weak signal

Figure 3.5 presents another case where signals are weak. Figure 3.5A shows that there are 7 weak uniform blocks in the true correlation matrix, but only three sample eigenvalues are slightly above the upper edge of the M-P law, as Figure 3.5B shows. Since the top sample eigenvalues deviate in a range characterized by the Tracy-Widom (T-

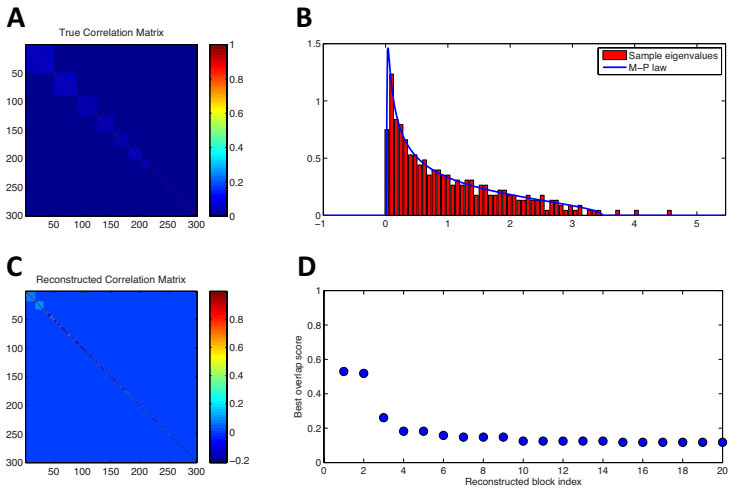


Figure 3.5: Simulation on a synthetic 300-by-400 Gaussian data matrix, whose underlying true correlation matrix has 7 uniform blocks of weak correlations. **A** shows the true correlation matrix. **B** shows the spectrum of the sample correlation matrix. Its top three eigenvalues are significantly above the upper edge of the M-P law. **C** and **D** respectively shows the reconstructed correlation matrix and the best overlap scores of the reconstructed blocks. Among the reconstructed blocks, two have medium sizes, with $\sim$ 0.5 best overlap scores; others are tiny with best overlap scores lower than 0.3.

W) distribution[52], we adopt the *eigenanalysis* method[83] to further justify the statistical significance of the sample eigenvalues. The method computes a statistics for each eigenvalue, which indicates its probability of being trivial. Table 3.1 lists these probabilities. It shows that the top three eigenvalues are informative, while others are trivial. This is consistent with the conclusion from the M-P law. However, three nontrivial eigenvalues do not necessarily mean that the algorithm can reconstruct three blocks perfectly, as signals partially mix with the noise when they are weak. As Figure 3.5C and D shows, the quality of the reconstruction is low. Specifically, only two reconstructed blocks have $\sim$ 0.5 best overlap scores; others have best overlap scores lower than 0.3. Although more iterations and longer runtime could potentially enhance the reconstruction, a more efficient method is necessary.

Eigenvector Index	Probability of being trivial
1	10^{-8}
2	10^{-8}
3	0.0042
4	0.9171
5	0.9996
6	1
7	1
⋮	⋮

Table 3.1: The probabilities of sample eigenvalues being trivial. They are calculated using the *Eigenanalysis* method [83].

3.4 Weak Signal Detection with Prior Information

At the failure of reconstructing weak signal, an intuitive question is whether we can incorporate prior information into the algorithm to enhance the detection. Since we aim to reconstruct correlation matrix with block structure, an example of prior information is the maximum number of blocks. Practically, the maximum number of blocks can be estimated in datasets where variables have been pre-categorized. Figure 3.6 shows two example datasets with variable pre-categorization. They are snippets of a yeast microarray dataset and a financial dataset. Many genes in the microarray dataset have annotations inferred from experiments, which reflect their basic functionalities. We can use the number of unique annotations in the dataset, or a larger number for safety, to estimate the maximum number of clusters in the dataset. Similarly, in the financial dataset, we can use the number of unique sectors or specialized industries to estimate the maximum number of clusters. Note that although the number of pre-defined categories reflects the number of *factors* in the dataset, both the gene annotation and company sector information might not represent the true classification of the variables.

Yeast microarray data

Gene ID	Annotation
YBR166C	TYR1 TYROSINE BIOSYNTHESIS
YOR357C	GRD19 SECRETION
YLR292C	SEC72 SECRETION
YGL112C	TAF60 TRANSCRIPTION
YIL118W	RHO3 CYTOSKELETON
YDL120W	YFH1 IRON HOMEOSTASIS
YHL025W	SNF6 TRANSCRIPTION
YGL248W	PDE1 PURINE METABOLISM
YIL146C	ECM37 CELL WALL BIOGENESIS
YJR106W	ECM27 CELL WALL BIOGENESIS
YNL272C	SEC2 SECRETION
YBR123C	TFC1 TRANSCRIPTION
YCR040W	ALPHA1 TRANSCRIPTION
YHR047C	AAP1 PROTEIN DEGRADATION
YMR055C	BUB2 CELL CYCLE, CHECKPOINT
YDR457W	TOM1 CELL CYCLE, G2/M
YKL201C	MNN4 PROTEIN GLYCOSYLATION
YDR311W	TFB1 TRANSCRIPTION

Financial data

Symbol	Sector	Industry
AAPL	Technology	Computer Manufacturing
GOOG	Technology	Computer Software: Programming, Data Processing
MSFT	Technology	Computer Software: Prepackaged Software
ORCL	Technology	Computer Software: Prepackaged Software
VOD	Public Utilities	Telecommunications Equipment
INTC	Technology	Semiconductors
AMZN	Consumer Services	Catalog/Specialty Distribution
CMCSA	Consumer Services	Television Services
CSCO	Technology	Computer Communications Equipment
QCOM	Technology	Radio And Television Broadcasting And Communications Equipment
CMCSK	Consumer Services	Television Services
GILD	Health Care	Biotechnology: Biological Products (No Diagnostic Substances)
AMOV	Public Utilities	Telecommunications Equipment
AMGN	Health Care	Biotechnology: Biological Products (No Diagnostic Substances)
EBAY	Miscellaneous	Business Services

Figure 3.6: Snippets of a yeast microarray dataset and a financial dataset. Although pre-categorization information, i.e. gene annotations in the microarray dataset and sector information in the financial dataset, do not necessarily reflect the genuine classifications of the variables, we can use them to estimate the maximum number of blocks.

For example, in the financial dataset, Amazon is pre-categorized as a consumer service company, while further analysis shows that its stock price is in fact correlated with the technology sector. Therefore, despite the pre-categorization, using a signal reconstruction algorithm to analyze the dataset is necessary.

When knowing the maximum number of blocks k a priori, the object of the algorithm becomes searching for C^* such that

$$\mathrm{C}^* = \mathrm{argmax}_{\mathrm{C}}\, \mathcal{L}(\mathrm{C} \,|\, \mathrm{X}, \text{Gaussian variables, C has at most } k \text{ blocks}). \tag{3.19}$$

Since $k \ll p$ in most problems, the search space of the algorithm becomes significantly smaller, and therefore it is easier for the algorithm to detect nontrivial signals.

We use two examples to demonstrate the efficacy of weak signal reconstruction after incorporating prior information. In the first example, we use the same synthetic dataset in Figure 3.5, where the true correlation matrix contains 7 weak uniform blocks. Figure 3.7A and B again show the true correlation matrix and spectrum. Without prior

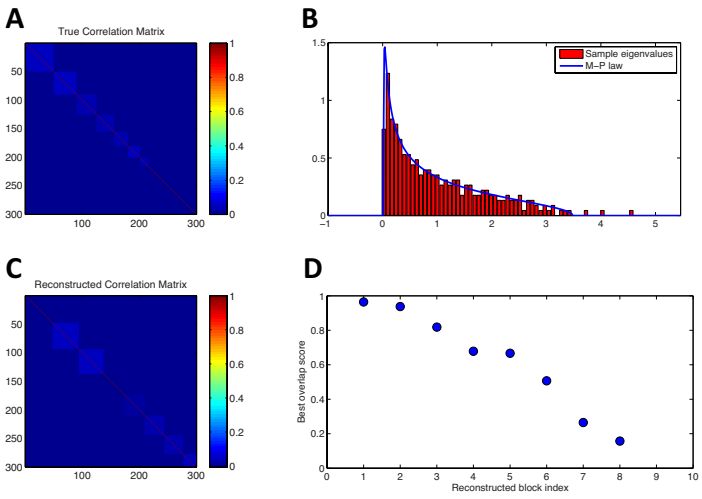


Figure 3.7: Signal reconstruction, with prior information incorporated in the search process, on dataset with weak uniform blocks. This is the same synthetic dataset used in Figure 3.5. **A** shows the true correlation matrix with seven uniform and weak blocks. **B** shows the spectrum of the dataset. There are only three nontrivial eigenvalues slightly above the upper edge of the M-P law. The maximum number of blocks is constrained to 8 in the search process. **C** and **D** show the reconstructed correlation matrix and the best overlap scores of the reconstructed blocks. The quality of the reconstruction is significantly improved comparing to Figure 3.5D. Particularly, the 4th and 5th blocks identify 60% of original blocks not identifiable by RMT.

information, the algorithm should search in a space where the correlation matrix could contain at most $p = 300$ blocks. However, if we know that the correlation matrix contains at most 7 nontrivial blocks a priori, we could constrain the search in a space where the correlation matrix contain at most $7 + 1 = 8$ blocks. The one extra block represent the group of uncorrelated variables. In the simulation, the algorithm executes the same number of iterations as before, and the results are presented in Figure 3.7C and D. The overlap scores of the reconstructed blocks are much higher than those identified without prior information. Particularly, the first three reconstructed blocks almost successfully identify the three nontrivial original blocks, and the 4th and 5th reconstructed blocks identify 60% of original blocks not identifiable by RMT.

In the second example, correlations within each of the 7 blocks are uniformly ran-

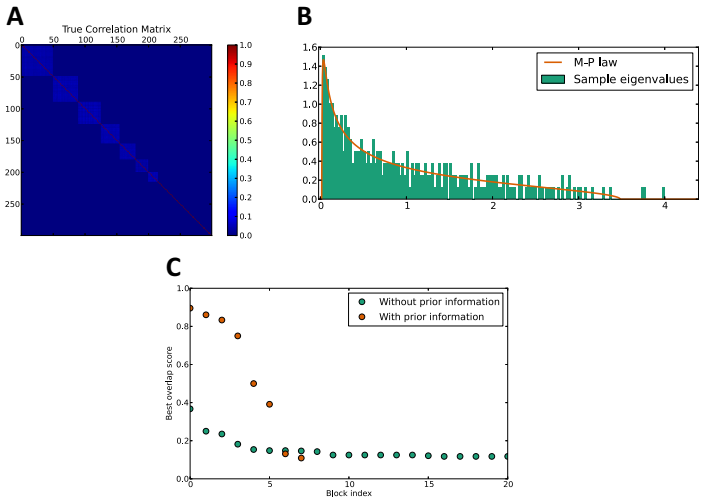


Figure 3.8: Signal reconstruction, with prior information incorporated in the search process, on dataset with weak nonuniform blocks. **A** shows the true correlation matrix. Correlations within blocks are uniformly random in $[0, 0.07]$. **B** shows the spectrum of the dataset. There are three nontrivial eigenvalues slightly above the upper edge of the M-P law. **C** presents the reconstruction results of the algorithm with and without prior information. It shows that incorporating prior information in the search significantly enhance the signal identification.

dom within $[0, 0.07]$. Figure 3.8A and B respectively shows the true correlation matrix and the spectrum. The maximum number of blocks is again constrained to 8 in the search process. Figure 3.8C shows that incorporating prior information in the algorithm significantly enhance the signal identification. Particularly, with a priori knowledge of the maximum number of blocks, the best overlap score of the 4th block, identified as trivial by the RMT, is more than 70%. This is another proof that incorporating prior information is an efficient method to detect signal below the RMT threshold.

3.5 Simulation on Financial Data

We also test the algorithm on real financial data to further demonstrate the method. The dataset consists of daily stock returns of $p = 486$ companies with the largest market capitals on NASDAQ and NYSE, in $n = 417$ trading days from Jan. 4, 2012 to Aug. 30, 2013. The stock lists are downloaded from the NASDAQ website[nas], which contain various information such as sector, specialized industry and market capital of the companies. The daily price data are fetched from Yahoo Finance[yah]. Figure 3.9 shows the daily return distributions of four stocks (green bars). They are closed to normal distribution (red curves), which is consistent with the efficient-market hypothesis[35]. On the other hand, due to extreme events, stock return distributions usually have heavier tail than normal distribution. People sometimes use Student's t-distribution and its skewed version as alternatives to model stock returns with heavy tail[14]. In our work however, we model the return series by normal distribution for simplicity.

Figure 3.10A shows the sample correlation matrix of the original dataset, where stocks are randomly ordered. Figure 3.10B is the sample correlation matrix after ordering the variables according to the clustering inferred by the algorithm. As we can see, clean cluster patterns surface in this reordered matrix. Figure 3.10C shows the reconstructed correlation matrix. Companies within small and strong blocks mostly belong to the same sector. For example, the 6th reconstructed block is composed of major banks, and the 8th block contains mostly semiconductor makers. This indicates that correlations between daily stock returns reflect specialized fields of the companies.

The clustering also produces several large blocks with weak average correlations. We can apply the algorithm again on these blocks to obtain fine-grained clustering. As an example, we use the return series of the 136 companies in the 1st reconstructed block as

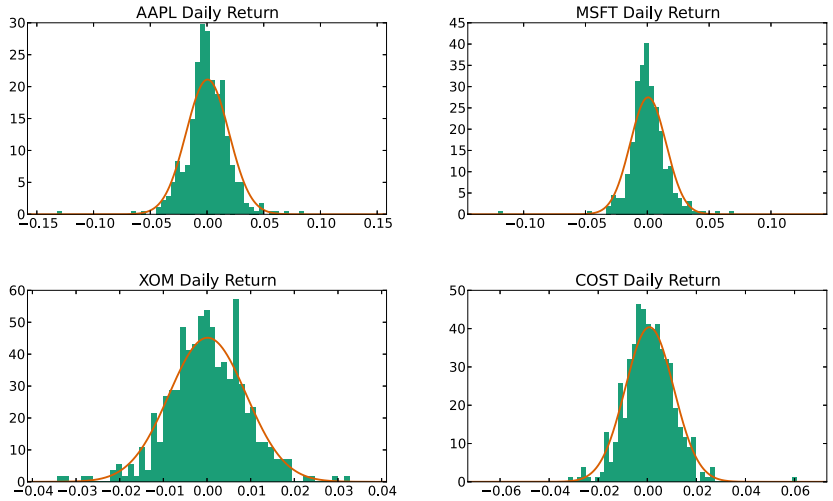


Figure 3.9: Green bars show the distributions of daily stock returns of Apple (AAPL), Microsoft (MSFT), Exxon Mobil (XOM) and Costco (COST). Red curves are normal distributions with the same mean and standard deviation as the returns. Stock returns are closed to Gaussian, consistent with the efficient-market hypothesis [35].

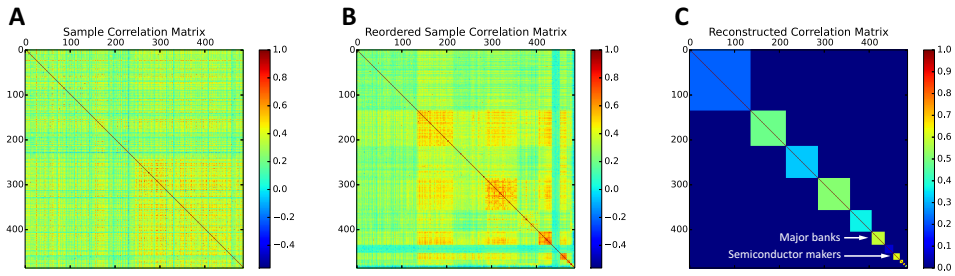


Figure 3.10: A shows the sample correlation matrix of the original dataset, where stocks are randomly ordered. **B** shows the sample correlation matrix where variables are ordered according to the clustering inferred by the algorithm. **C** shows the reconstructed correlation matrix. Companies within small and strong blocks mostly specialized in the same field. For example, the 6th reconstructed block is composed of major banks, and companies in the 8th block are mostly semiconductor makers.

a new dataset for *reclustering*. Figure 3.11A shows the sample correlation matrix with random variable order, and Figure 3.11B shows the spectrum of the new dataset. There are three eigenvalues above the upper edge of the M-P law, indicating that there are only three significant blocks in the dataset. Figure 3.11C and D respectively shows the algorithmic-reordered sample correlation matrix and the reconstructed correlation matrix, where the algorithm does not incorporate any prior information, or the search is *unsupervised*. They show that the algorithm is only able to detect three mini blocks, each has less than 5 companies. This is consistent with the RMT inference. Note that the 136 companies in this dataset belong to 58 unique industries. We therefore execute the algorithm again by constraining the maximum number of blocks to 58, or the search is *supervised*. Figure 3.12 shows the result. A major block is identified after incorporating prior information. Companies within this block are mostly pharmaceutical and biotech companies, reflecting the genuineness of this cluster. This adds a real-data example demonstrating that incorporating prior information is an efficient method to identify weak signals.

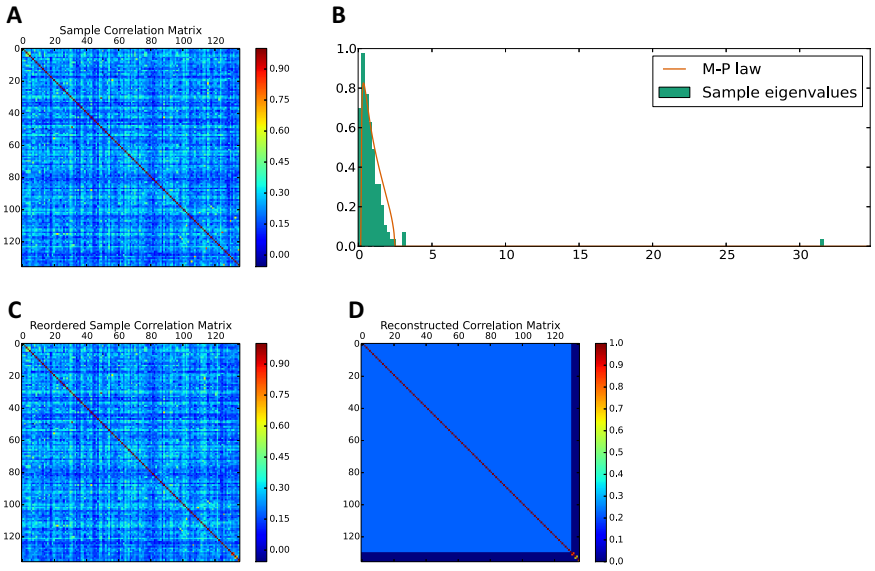


Figure 3.11: Unsupervised reclustering result on the 136 companies in the 1st reconstructed block of the initial clustering. **A** shows the sample correlation matrix. **B** shows the spectrum of the dataset, where three significant eigenvalues are above the upper edge of the M-P law. **C** and **D** respectively shows the reordered sample correlation matrix and the reconstructed correlation matrix. The algorithm detect three mini blocks, each has less than five companies. This is consistent with the RMT inference.

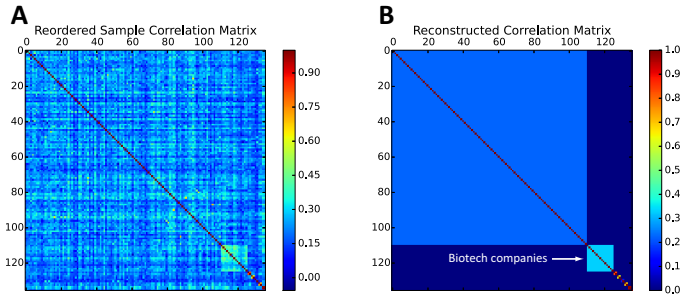


Figure 3.12: Supervised reclustering result on the 136 companies in the 1st reconstructed block of the initial clustering. The maximum number of blocks is constrained to 58, which is the number of unique industries these companies belong to. **A** shows the sample correlation matrix where variables are ordered according to the clustering inferred by the algorithm. **B** shows the reconstructed correlation matrix. In the same number of iterations, the algorithm identifies an extra block composed of pharmaceutical and biotech companies.

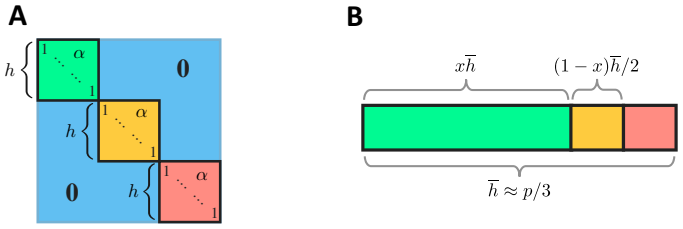


Figure 3.13: Schema of the homogeneous model when the number of blocks $k = 3$. **A** shows the true correlation matrix, in which each variable belongs to one of the 3 uniform blocks. All of them have size h and correlation α. **B** shows the composition of a reconstructed block. In the homogeneous model, since $k \ll p$, the size $\overline{h}$ of each reconstructed block is approximately p/k; variables belonging to a specific original block take the dominant fraction $x \in [0, 1]$ of this block, while variables belonging to other original blocks equally share the rest.

3.6 Quantify Prior Information

In this section, we use a homogeneous model to quantify to what extent the prior knowledge of maximum number of blocks enhances signal detection. In this model, we assume that a) each variable belong to one of the k uniform blocks, where $k \ll p$, and b) blocks have equal size h and strength α. Figure 3.13A shows the true correlation matrix of such a model when $k = 3$. Under these homogeneous assumptions, when the number of blocks k is known a priori, each reconstructed block during the search process has size $\overline{h} \approx p/k$; variables belonging to a specific original block take the dominant fraction of this reconstructed block, while variables belonging to other original blocks equally share the rest of the block. Figure 3.13B shows a schema of the composition of a reconstructed block when $k = 3$, where $x \in [0, 1]$ represents the fraction of the variables belonging to a specific original block.

Since all original blocks are equivalent to each other, and assuming k is known a priori, we can conveniently study the energy U of an individual reconstructed block as

a representative of the energy of the whole reconstructed correlation matrix:

$$\begin{aligned} U &= \overline{h} + (1-\overline{\alpha})^{\overline{h}-1}[(\overline{h}-1)\overline{\alpha}+1] \\ &\approx \frac{p}{k} + (1-\overline{\alpha})^{p/k-1}[(\frac{p}{k}-1)\overline{\alpha}+1], \end{aligned} \tag{3.20}$$

where $\overline{\alpha}$ is approximated by the average correlation within the reconstructed block. When $n \to \infty$ and $p/n \to 0$, $\overline{\alpha}$ is an analytical function of x:

$$\overline{\alpha}(x) = \frac{p\alpha(k-2)}{(k-1)(p-k)}x^2 + \frac{\alpha(p+k-k^2)}{(k-1)(p-k)}, \tag{3.21}$$

so that U is also an analytical function of x. Figure 3.14A shows the energy U as a function of x when $k = 5, \alpha = 0.1, p = 500$ and $n \to \infty$. It shows that the energy reaches its minimum when $x = 1$, or the block can be perfectly reconstructed. When the number of samples n is finite, U is no longer an analytical function of x. In this case, we can depict U as a function of x by generating synthetic dataset according to the true correlation matrix and randomly selecting variables into the reconstructed block for a series of x. Figure 3.14B and C respectively shows U as a function of x for $\alpha = 0.1$ and 0.005, when $k = 5, p = 500$ and $n = 200$. As expected, they show that the energy landscape is rougher when the signal becomes weaker. Since theoretically the simulated annealing algorithm is able to locate the minimum given sufficient time, if $\text{argmin}_x(U) = 1$, the block can be perfectly reconstructed no matter how rough the energy landscape is. Therefore, one way to quantify the detection power of knowing k a priori is to investigate how small α could be while the equation $\text{argmin}_x(U) = 1$ still holds.

The green curve in Figure 3.15A shows an average of $\text{argmin}_x(U)$ over 100 simulations as a function of α when $k = 5, p = 500$ and $n = 200$. Solid curves in Figure 3.15B

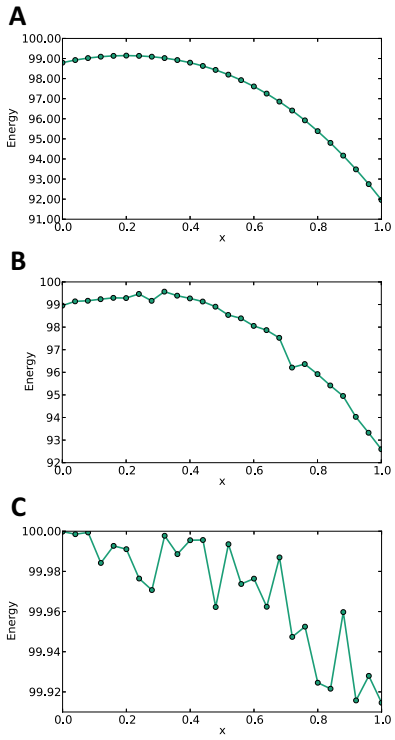


Figure 3.14: The energy U of a single reconstructed block as a function of $x \in [0, 1]$ for $k = 5$, $p = 500$. **A** shows the curve when $\alpha = 0.1$ and $n = \infty$. In this case, U is an analytical function of x, and $\text{argmin}_x(U) = 1$ as expected. **B** and **C** show two finite sample cases when $n = 200$, and $\alpha = 0.1$ and 0.005 respectively. The energy landscape is rougher when the signal α becomes weaker. One way to quantify the detection power of knowing k a priori is to investigate how small α could be while the equation $\text{argmin}_x(U) = 1$ still holds.

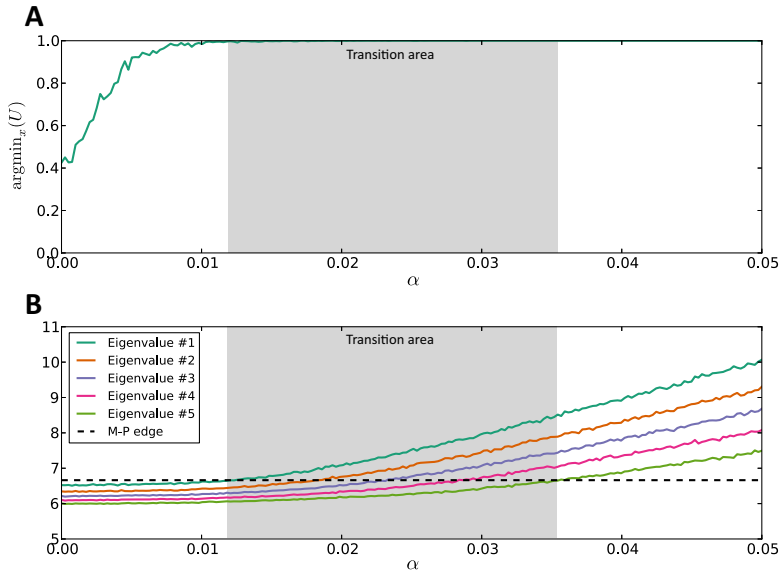


Figure 3.15: A shows an average of $\mathrm{argmin}_x(U)$ over 100 simulations as a function of α when $k = 5$, $p = 500$ and $n = 200$. **B** shows the average top five eigenvalues of the sample correlation matrix in solid curves, as well as the upper edge of the M-P law in dashed line. Although information gradually loses in the gray *transition area*, where eigenvalues vanish into the bulk of the M-P law one by one, $\mathrm{argmin}_x(U)$ stays at 1. This infers that when the RMT based detection partially fails, signals can still be fully reconstructed given the prior knowledge of k. On the left hand side of the transition area where all nontrivial eigenvalues vanish, there is a region where $\mathrm{argmin}_x(U)$ is equal or closed to 1, which indicates that it is still possible to reconstruct all signals when the whole dataset is determined to be trivial by RMT.

shows the corresponding average top five eigenvalues. These five eigenvalues vanish into the bulk of the M-P law one by one in the gray *transition area*, which reflects increasing information loss; meanwhile $\mathrm{argmin}_x(U)$ stays at 1, indicating potential perfect signal identification if knowing the exact number of blocks $k = 5$ a priori. On the left hand side of the transition area where all nontrivial eigenvalues vanish, there is a region where $\mathrm{argmin}_x(U)$ is equal or closed to 1. These infer that when knowing the exact number of blocks a priori, all signals can be theoretically reconstructed when the RMT based detection partially or completely fails.

3.7 Summary

To summarize, this chapter presents a methodology based on simulated annealing to efficiently reconstruct the optimal true correlation matrix of a particular pattern for multivariate datasets. The algorithm is superior in that it is natural to incorporate prior knowledge into the search process, so that signals below the Random Matrix Theory threshold become detectable. Simulations on both synthetic and real financial datasets demonstrate the efficacy of the method. A homogeneous model is also proposed to explain the dramatic increase in signal detection accuracy when incorporating the knowledge of the number of blocks.

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it.

Steve Jobs

4

Conclusion

This dissertation elaborates the applications of state-of-the-art computation techniques and data analysis algorithms in three representative physical and biological problems: assembling DNA pieces, optimizing self-assembly yield and identifying correlations from multivariate datasets. In-depth study of Sequencing by Hybridization algorithm demonstrates how a variant of the classical reconstruction algorithm can significantly extend the length of target that can be sequenced with standard oligonucleotide

probes, which creates new opportunities to use SBH along with microfluidic technology in the next generation sequencing. Simulations using GPU computing show how controlling the short-ranged interactions between particles and controlling the concentrations optimize the self-assembly yield of a desired structure, and nonequilibrium behavior when optimizing concentrations is uncovered by leveraging the enormous computation capacity of GPUs. At last, a methodology to incorporate existing categorization information into the search process to efficiently reconstruct the optimal true correlation matrix for multivariate datasets is introduced. Simulations on both synthetic and real financial datasets show that it is capable to detect signals below the Random Matrix Theory threshold.

As I point out at the beginning, although these three problems are superficially independent, intrinsically they share the same spirit of leveraging computation and data analysis techniques to tackle optimization problems, and outperform theoretical boundary when incorporating prior information into the computation. As the data collection momentum continues to grow exponentially fast, it is predictable that in the near future massive computation techniques and data analysis algorithms will become an essential tool to tackle analytically complicated scientific problems.

References

[yah] http://finance.yahoo.com.

[vis] http://people.seas.harvard.edu/∼qin/visrepeat/ (make sure the '∼' symbol is correctly input).

[gnu] http://www.gnubio.com.

[nas] http://www.nasdaq.com/screening/company-list.aspx.

[5] Abate, A., Agresti, J., & Weitz, D. (2010a). Microfluidic sorting with high-speed single-layer membrane valves. *Appl Phys Lett*, 96, 203509.

[6] Abate, A., Hung, T., Mary, P., Agresti, J., & Weitz, D. (2010b). High-throughput injection with microfluidics using picoinjectors. *Proc Natl Acad Sci U S A*, 107(45), 19163–19166.

[7] Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769), 503–511.

[8] Anderson, J. A., Lorenz, C. D., & Travesset, A. (2008). General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10), 5342–5359.

[9] Arkus, N. (2009). *Theoretical approaches to self-assembly and biology*. PhD thesis, Harvard University, Cambridge, MA.

[10] Arkus, N., Manoharan, V. N., & Brenner, M. P. (2009). Minimal energy clusters of hard spheres with short range attractions. *Physical review letters*, 103(11), 118303.

[11] Arratia, R., Martin, D., Reinert, G., & Waterman, M. (1996). Poisson process approximation for sequence repeats, and sequencing by hybridization. *J Comput Biol*, 3(3), 425–463.

[12] Bai, Z. & Silverstein, J. W. (2010). *Spectral analysis of large dimensional random matrices*. Springer.

[13] Bains, W. & Smith, G. (1988). A novel method for nucleic acid sequence determination. *J Theor Biol*, 135(3), 303–307.

[14] Bauwens, L. & Laurent, S. (2005). A new class of multivariate skew densities, with application to generalized autoregressive conditional heteroscedasticity models. *Journal of Business & Economic Statistics*, 23(3).

[15] Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer Jr, E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., & Tasumi, M. (1977). The protein data bank: a computer-based archival file for macromolecular structures. *Journal of molecular biology*, 112(3), 535–542.

[16] Biancaniello, P. L., Kim, A. J., & Crocker, J. C. (2005). Colloidal interactions and self-assembly using dna hybridization. *Physical review letters*, 94(5), 058302.

[17] Binder, K. & Heermann, D. W. (2010). *Monte Carlo simulation in statistical physics: an introduction*. Springer.

[18] Blazewicz, J., Formanowicz, P., Kasprzak, M., Markiewicz, W., & Weglarz, J. (2000). Tabu search for dna sequencing with false negatives and false positives. *Eur J Oper Res*, 125(2), 257–265.

[19] Błażewicz, J., Formanowicz, P., Kasprzak, M., Markiewicz, W. T., & W glarz, J. (1999). Dna sequencing with positive and negative errors. *Journal of Computational Biology*, 6(1), 113–123.

[20] Blazewicz, J., Glover, F., & Kasprzak, M. (2005). Evolutionary approaches to dna sequencing with errors. *Ann Oper Res*, 138(1), 67–78.

[21] Blazewicz, J. & Kasprzak, M. (2003). Complexity of dna sequencing by hybridization. *Theor Comput Sci*, 290(3), 1459–1473.

[22] Blum, C., Valles, M., & Blesa, M. (2008). An ant colony optimization algorithm for dna sequencing by hybridization. *Comput Oper Res*, 35(11), 3620–3635.

[23] Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence*. Oxford.

[24] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., & Skadron, K. (2008). A performance study of general-purpose applications on graphics processors using cuda. *Journal of parallel and distributed computing*, 68(10), 1370–1380.

[25] Cohen, J. & Molemaker, M. J. (2009). A fast double precision cfd code using cuda. *Parallel Computational Fluid Dynamics: Recent Advances and Future Directions*, (pp. 414–429).

[26] Corrigan, A., Camelli, F. F., Löhner, R., & Wallin, J. (2011). Running unstructured grid-based cfd solvers on modern graphics hardware. *International Journal for Numerical Methods in Fluids*, 66(2), 221–229.

[27] Cutler, D., Zwick, M., Carrasquillo, M., Yohn, C., Tobin, K., Kashuk, C., Mathews, D., Shah, N., Eichler, E., Warrington, J., et al. (2001). High-throughput variation detection and genotyping using microarrays. *Genome Res*, 11(11), 1913–1925.

[28] Dahirel, V., Shekhar, K., Pereyra, F., Miura, T., Artyomov, M., Talsania, S., Allen, T. M., Altfeld, M., Carrington, M., Irvine, D. J., et al. (2011). Coordinate linkage of hiv evolution reveals regions of immunological vulnerability. *Proceedings of the National Academy of Sciences*, 108(28), 11530–11535.

[29] Drmanac, R., Drmanac, S., Strezoska, Z., Paunesku, T., Labat, I., Zeremski, M., Snoddy, J., Funkhouser, W., Koop, B., Hood, L., et al. (1993). Dna sequence determination by hybridization: a strategy for efficient large-scale sequencing. *Science*, 260(5114), 1649–1652.

[30] Drmanac, R., Labat, I., Brukner, I., & Crkvenjakov, R. (1989). Sequencing of megabase plus dna by hybridization: theory of the method. *Genomics*, 4(2), 114–128.

[31] Dyer, M., Frieze, A., & Suen, S. (1994). The probability of unique solutions of sequencing by hybridization. *J Comput Biol*, 1(2), 105–110.

[32] Edelman, A. & Rao, N. R. (2005). Random matrix theory. *Acta Numerica*, 14, 233–297.

[33] Endo, T. (2004). Probabilistic nucleotide assembling method for sequencing by hybridization. *Bioinformatics*, 20(14), 2181–2188.

[34] Ericson, C. (2005). *Real-time collision detection*. Taylor & Francis US.

[35] Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, 38(1), 34–105.

[36] Feynman, R. P. (1982). Simulating physics with computers. *International journal of theoretical physics*, 21(6), 467–488.

[37] Frieze, A., Preparata, F., & Upfal, E. (1999). Optimal reconstruction of a sequence from its probes. *J Comput Biol*, 6(3-4), 361–368.

[38] Gibbs, J. W. (2010). *Elementary principles in statistical mechanics: developed with especial reference to the rational foundation of thermodynamics*. Cambridge University Press.

[39] Glotzer, S. C. (2004). Some assembly required. *Science*, 306(5695), 419–420.

[40] Groot, R. D. & Warren, P. B. (1997). Dissipative particle dynamics: bridging the gap between atomistic and mesoscopic simulation. *Journal of Chemical Physics*, 107(11), 4423.

[41] Hacia, J. G. (1999). Resequencing and mutational analysis using oligonucleotide microarrays. *Nature genetics*, 21, 42–47.

[42] Hager, G. & Wellein, G. (2010). *Introduction to high performance computing for scientists and engineers*. CRC Press.

[43] Halverson, J. D. & Tkachenko, A. V. (2013). Dna-programmed mesoscopic architecture. *Physical Review E*, 87(6), 062310.

[44] Hamada, T., Narumi, T., Yokota, R., Yasuoka, K., Nitadori, K., & Taiji, M. (2009). 42 tflops hierarchical n-body simulations on gpus with applications in both astrophysics and turbulence. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (pp.62).: ACM.

[45] Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1), 97–109.

[46] Heath, M. T. (1997). *Scientific computing*. McGraw-Hill New York.

[47] Hillier, L. W., Reinke, V., Green, P., Hirst, M., Marra, M. A., & Waterston, R. H. (2009). Massively parallel sequencing of the polyadenylated transcriptome of c. elegans. *Genome research*, 19(4), 657–666.

[48] Hood, L. (2003). Systems biology: integrating technology, biology, and computation. *Mechanisms of ageing and development*, 124(1), 9–16.

[49] Hoogerbrugge, P. & Koelman, J. (1992). Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *EPL (Europhysics Letters)*, 19(3), 155.

[50] Hormoz, S. & Brenner, M. P. (2011). Design principles for self-assembly with short-range interactions. *Proceedings of the National Academy of Sciences*, 108(13), 5193–5198.

[51] Johnson, M. E. & Hummer, G. (2011). Nonspecific binding limits the number of proteins in a cell and shapes their interaction networks. *Proceedings of the National Academy of Sciences*, 108(2), 603–608.

[52] Johnstone, I. M. (2001). On the distribution of the largest eigenvalue in principal components analysis. *Annals of statistics*, (pp. 295–327).

[53] Johnstone, I. M. (2006). High dimensional statistical inference and random matrices. *arXiv preprint math/0611589*.

[54] Jolliffe, I. (2005). *Principal component analysis*. Wiley Online Library.

[55] Jones, J. (1924). On the determination of molecular fields. ii. from the equation of state of a gas. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(738), 463–477.

[56] Kern, N. & Frenkel, D. (2003). Fluid–fluid coexistence in colloidal systems with short-ranged strongly directional attraction. *The Journal of chemical physics*, 118, 9882.

[57] Khrapko, K., P Lysov, Y., Khorlyn, A., Shick, V., Florentiev, V., & Mirzabekov, A. (1989). An oligonucleotide hybridization approach to dna sequencing. *FEBS letters*, 256(1), 118–122.

[58] Kicinger, R., Arciszewski, T., & Jong, K. D. (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23), 1943–1978.

[59] Kirkpatrick, S., Jr., D. G., & Vecchi, M. P. (1983). Optimization by simmulated annealing. *science*, 220(4598), 671–680.

[60] Kutz, J. N. (2013). *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press.

[61] Laloux, L., Cizeau, P., Bouchaud, J.-P., & Potters, M. (1999). Noise dressing of financial correlation matrices. *Physical Review Letters*, 83(7), 1467.

[62] Leiserson, C. E., Rivest, R. L., Stein, C., & Cormen, T. H. (2001). *Introduction to algorithms*. The MIT press.

[63] Levin, S. A., Grenfell, B., Hastings, A., & Perelson, A. S. (1997). Mathematical and computational challenges in population biology and ecosystems science. *Science*, 275(5298), 334–343.

[64] Levy, E. D., Pereira-Leal, J. B., Chothia, C., & Teichmann, S. A. (2006). 3d complex: a structural classification of protein complexes. *PLoS computational biology*, 2(11), e155.

[65] Lin, Z., Hahm, T. S., Lee, W., Tang, W. M., & White, R. B. (1998). Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, 281(5384), 1835–1837.

[66] Lynch, C. (2008). Big data: How do your data grow? *Nature*, 455(7209), 28–29.

[67] Marčenko, V. A. & Pastur, L. A. (1967). Distribution of eigenvalues for some sets of random matrices. *Sbornik: Mathematics*, 1(4), 457–483.

[68] Martinez-Veracoechea, F. J., Mladek, B. M., Tkachenko, A. V., & Frenkel, D. (2011). Design rule for colloidal crystals of dna-functionalized particles. *Physical review letters*, 107(4), 045902.

[69] Meng, G., Arkus, N., Brenner, M. P., & Manoharan, V. N. (2010). The free-energy landscape of clusters of attractive hard spheres. *Science*, 327(5965), 560–563.

[70] Murugan, A., Zou, J., & Brenner, M. P. (2013). Principles for robust self-assembly of heterogeneous structures at finite concentration. *manuscript under review*.

[71] Nadakuditi, R. R. & Newman, M. E. (2012). Graph spectra and the detectability of community structure in networks. *Physical review letters*, 108(18), 188701.

[72] Nadakuditi, R. R. & Newman, M. E. (2013). Spectra of random graphs with arbitrary expected degrees. *Physical Review E*, 87(1), 012803.

[73] Needleman, S. & Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3), 443–453.

[74] Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with cuda. *Queue*, 6(2), 40–53.

[75] Nickolls, J. & Dally, W. J. (2010). The gpu computing era. *IEEE micro*, 30(2), 56–69.

[76] NVIDIA (2012). Nvidia cuda c programming guide.

[77] Nyquist, H. (1928). Thermal agitation of electric charge in conductors. *Physical review*, 32(1), 110–113.

[78] Olson, G. B. (1997). Computational design of hierarchically structured materials. *Science*, 277(5330), 1237–1242.

[79] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). Gpu computing. *Proceedings of the IEEE*, 96(5), 879–899.

[80] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., & Purcell, T. J. (2007). A survey of general-purpose computation on graphics hardware. In *Computer graphics forum*, volume 26 (pp. 80–113).: Wiley Online Library.

[81] Park, S. Y., Lytton-Jean, A. K., Lee, B., Weigand, S., Schatz, G. C., & Mirkin, C. A. (2008). Dna-programmable nanoparticle crystallization. *Nature*, 451(7178), 553–556.

[82] Parris, K. D., Lin, L., Tam, A., Mathew, R., Hixon, J., Stahl, M., Fritz, C. C., Seehra, J., & Somers, W. S. (2000). Crystal structures of substrate binding to< i> bacillus subtilis</i> holo-(acyl carrier protein) synthase reveal a novel trimeric arrangement of molecules resulting in three active sites. *Structure*, 8(8), 883–895.

[83] Patterson, N., Price, A. L., & Reich, D. (2006). Population structure and eigenanalysis. *PLoS genetics*, 2(12), e190.

[84] Pevzner, P. (1989). 1-tuple dna sequencing: computer analysis. *J Biomol Struct Dyn*, 7(1), 63–73.

[85] Pickrell, J. K., Marioni, J. C., Pai, A. A., Degner, J. F., Engelhardt, B. E., Nkadori, E., Veyrieras, J.-B., Stephens, M., Gilad, Y., & Pritchard, J. K. (2010). Understanding mechanisms underlying human gene expression variation with rna sequencing. *Nature*, 464(7289), 768–772.

[86] Plerou, V., Gopikrishnan, P., Rosenow, B., Amaral, L. A. N., & Stanley, H. E. (1999). Universal and nonuniversal properties of cross correlations in financial time series. *Physical Review Letters*, 83(7), 1471.

[87] Qin, Y., Schneider, T. M., & Brenner, M. P. (2012). Sequencing by hybridization of long targets. *PloS one*, 7(5), e35819.

[88] Rothemund, P. W. (2000). Using lateral capillary forces to compute by self-assembly. *Proceedings of the National Academy of Sciences*, 97(3), 984–989.

[89] Sanders, J. & Kandrot, E. (2010). *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional.

[90] Santhanam, M. & Patra, P. K. (2001). Statistics of atmospheric correlations. *Physical Review E*, 64(1), 016102.

[91] Scherl, H., Keck, B., Kowarschik, M., & Hornegger, J. (2007). Fast gpu-based ct reconstruction using the common unified device architecture (cuda). In *Nuclear Science Symposium Conference Record, 2007. NSS'07. IEEE*, volume 6 (pp. 4464–4466).: IEEE.

[92] Stone, J. E., Vandivort, K. L., & Schulten, K. (2013). Gpu-accelerated molecular visualization on petascale supercomputing platforms. In *Proceedings of the 8th International Workshop on Ultrascale Visualization* (pp.6).: ACM.

[93] Stone, S. S., Haldar, J. P., Tsao, S. C., Hwu, W.-m., Sutton, B. P., Liang, Z.-P., et al. (2008). Accelerating advanced mri reconstructions on gpus. *Journal of Parallel and Distributed Computing*, 68(10), 1307–1318.

[94] Swope, W. C., Andersen, H. C., Berens, P. H., & Wilson, K. R. (1982). A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of Chemical Physics*, 76, 637.

[95] Tulino, A. M. & Verdú, S. (2004). *Random matrix theory and wireless communications*, volume 1. Now Publishers Inc.

[96] Valignat, M.-P., Theodoly, O., Crocker, J. C., Russel, W. B., & Chaikin, P. M. (2005). Reversible self-assembly and directed assembly of dna-linked micrometer-sized colloids. *Proceedings of the National Academy of Sciences of the United States of America*, 102(12), 4225–4229.

[97] Van Laarhoven, P. J. & Aarts, E. H. (1987). *Simulated annealing*. Springer.

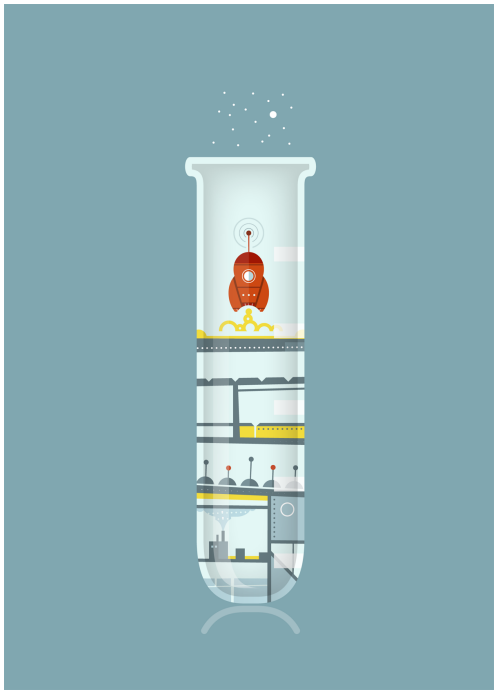
[98] Verlet, L. (1967). Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1), 98.

[99] Vetter, J. S. (2013). *Contemporary High Performance Computing: From Petascale Toward Exascale*. CRC Press.

[100] Whitesides, G. M. & Grzybowski, B. (2002). Self-assembly at all scales. *Science*, 295(5564), 2418–2421.

[101] Whitesides, G. M., Mathias, J. P., & Seto, C. T. (1991). *Molecular self-assembly and nanochemistry: a chemical strategy for the synthesis of nanostructures*. Technical report, DTIC Document.

[102] Winfree, E., Liu, F., Wenzler, L. A., & Seeman, N. C. (1998). Design and self-assembly of two-dimensional dna crystals. *Nature*, 394(6693), 539–544.



This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, "Science Experiment 02", was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD thesis with this look and feel has been released under the permissive MIT (XII) license, and can be found online at github.com/suchow/Dissertate or from its author, Jordan Suchow, at suchow@post.harvard.edu.